# SAMPLE EDGE OFFSET COMPENSATION FOR HEVC BASED 3D VIDEO CODING

*Qin Yu [1], Ying Chen[2], Li Zhang[2], Siwei Ma[1]*

[1]Institute of Digital Media, Peking University
{qyu, swma}@pku.edu.cn
[2]Qualcomm Inc. 5775 Morehouse Dr. San Diego, CA, USA 92121
{cheny, lizhang}@qti.qualcomm.com

## ABSTRACT

In this paper, an in-loop sample edge offset compensation (SEOC) framework is proposed for High Efficiency Video Coding (HEVC) based 3D video (3D-HEVC) coding. The framework targets at improving the reconstruction quality of the depth images, especially in edge areas. In a typical 3DV system, depth images are used for synthesizing the virtual views, therefore preserving high quality depth images, especially the edge information is important. However, the ringing artifacts may be introduced at the depth edge areas due to the compression distortion of the depth images. The SEOC framework resolves this problem, after reconstruction, by identifying each edge pixel and enhancing it of the full depth image with values coded in the bitstream. Experimental results demonstrate that, compared with the original 3D-HEVC, the proposed algorithm can achieve about 6% bitrate saving on average.

***Index Terms***—3D-HEVC, depth coding, depth edge filtering, in-loop filter

## 1. INTRODUCTION

3D video applications such as 3DTV and free view-point video become more and more popular. Currently, Joint Collaboration Team on 3D Video Coding (JCT-3V) of VCEG and MPEG is developing a 3DV standard based on HEVC, namely 3D-HEVC [1]. In 3D-HEVC multiple texture views are coded, in addition the corresponding depth images of these views are coded. After decoding the 3D-HEVC bitstreams and reconstructing the coded texture and depth views, additional virtual views can be rendered by decoded textures and depth images based on the technique, such as of depth image based rendering (DBIR) [2].

Since depth images play an important role in depth based rendering, many methods have been proposed to suppress the noise and enhance the reconstructed quality of depth images. These algorithms can be classified into two categories: the first is to remove the blocking artifacts such as deblocking filter; the other is to preserve the sharp edge such as bilateral filter or trilateral filter.

In [3], an improved in-loop filter based on H.264/AVC deblocking filter was proposed for depth image coding, in which a 4-tap filter replaced the one in H.264/AVC, but it does not perform well at low bitrate. In [4], a context adaptive in-loop filter is proposed to preserve the depth details and improve the quality of synthesized views. Similar to the deblocking filter, the transform unit (TU) information is utilized and they are classified into three categories which is flat, directional and textureless. These methods are effective in blocking artifacts removal.

However, except for the blocking artifacts, ringing artifacts also exist around the sharp edges in the reconstructed depth image. The above mentioned in-loop filters works well in removing the blocking artifacts; Nguyen et al. [5] presented the weighted mode filtering on the reconstructed depth image to suppress the remaining coding artifacts. In [6], a bilateral filter is proposed taking both blocking artifacts removal and sharp edges preserving into account. The proposed filter is simple and effective, especially at low bitrate. In [7], an adaptive bilateral filtering (ABF) scheme is proposed in which filter parameters can be determined region by region in the encoder and transmitted to the decoder. Based on the bilateral filter, Jageret et al. [8] proposed a new in-loop filter technique termed as median trilateral loop filter, which can reduce the ringing artifacts and well align the object boundaries. Liu et al. [9] also presented a trilateral filter by using the similarity of depth image and corresponding texture image. But the performance isn't stable if there are few similar pixels around the current pixel. What's more, this method may cause blur around edges because it uses the weighted average techniques. To address this first issue, Zhong et al. [10] introduced a robust local binary pattern (LBP) guided depth image filter in which only the local neighborhood samples in the same object of the current pixel can be filtered. For the second issue, a depth image boundary filtering containing L0-norm minimization is proposed in [11]. With the proposed method, coding artifacts can be eliminated, and sharp edges can be preserved and no other artifacts are brought into the processed image. Based on ABF, an adaptive joint trilateral filter is further introduced in [12] to suppress the noise and sharpening the edges simultaneously.

All these proposed algorithms aim at improving the quality or visual quality of depth images. However, depth images with higher quality don't necessarily lead to higher quality of synthesized views. Methods for view synthesis, such as DIBR, are sensitive to the depth edges, while conventional block based video coding schemes blur the edges, even if the depth images are coded with high bit rates. So the edge pixels of a depth block need to be corrected; however, the existing residue coding method doesn't consider much on this issue.

In this paper, we propose an in-loop sample offset compensation (SEOC) framework. In this framework, there are two compensation modes: adaptive depth edge filtering (ADEF) and sample edge offset (SEO). The reconstructed samples are adaptively classified into different categories. For ADEF, the pixels are filtered based only on the values of the neighboring samples; while for SEO, the offset is compressed using Depth Lookup Table (DLT) [13] and transmitted explicitly to be added to the relevant reconstructed samples. To select a better mode for the current Largest Coding Unit (LCU) [14], view synthesis optimization (VSO) [15] is used taking both the quality of both the depth images and the synthesized views into consideration. Besides the LCU-level decision, frame-level decision is also taken into consideration to achieve a better rate distortion performance. The rest of this paper is organized as follows. In Section 2, we first analyze the characteristics of depth image and the two specific coding tools in details, including DLT and VSO, which will be utilized later. In Section 3, we describe the SEOC framework, including the encoder design to support SEOC. Experimental results are provided in Section 4, and Section 5 concludes the paper.

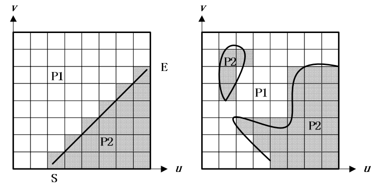
## 2. DEPTH IMAGE CODING

Depth image has many characteristics that make it different from texture image. Firstly, the depth images are characterized by piecewise smooth regions, which are bounded by sharp edges describing depth discontinuities along object boundaries. And only a small amount of different depth levels occur due to the strong quantization. Moreover, depth images are not viewed by people directly. It is the synthesized views rendered by depth images that are viewed. So the goal of depth compressing is not only preserving better quality of depth images themselves but also the synthesized views. Because of the particular characteristics of depth images, it's not efficient to compress them using the compression techniques for the textures. For example, if we use 8 bits to represent the depth value, there exists redundancy since only a small account of depth levels occur. Ringing artifacts and blocking artifacts may be introduced because of transform and quantization. Thus, sharp edges and smooth regions cannot be preserved, and finally result in bad quality in synthesized view.

In 3D-HEVC, techniques that are designed according to the characteristics of depth image have been proposed, in which two important encoding techniques are Depth Modeling Mode (DMM) and View Synthesis Optimization (VSO) [16]. DMM is utilized to better represent the sharp object borders and constant object areas. In addition, DLT is utilized to reduce redundancy in depth value representation by mapping the discontinuous depth value to continuous table index. VSO is used to select better mode in the point of better synthesized view. In the next subsections, we will introduce these two techniques in detail.

### 2.1. Depth modeling mode

To better preserving the depth information, two depth intra modes as shown in Figure 1 are developed and added into the depth coding process. A block is approximated by a model that partitions it into two parts, namely P1 and P2. To partition the prediction unit into two parts, wedgelet which is signalled by a straight line and Contour which can be arbitrary shaped are used as illustrate in Figure 1 (a) and (b) respectively. Similar to intra prediction modes, a residual representing the difference between the prediction and the original signal can be transmitted.



(a) Wedgelet (b) Contour

**Fig. 1.** Depth modeling mode.

Depth image is usually represented as a grayscale image with 8 bits per pixel. However, in many cases, the dynamic range of the depth values is narrower than the full range represented by 8 bits. The DLT utilizes this property of the depth image. In the encoder, a dynamic depth lookup table is constructed by analyzing a certain number of frames of the input sequence. This DLT is used during the coding process to reduce the effective signal bit-depth of the residual signal.

At the encoder side, residual index ires to be coded into the bitstream, is calculated by the index difference of the original depth value and the predicted depth value, as given by:

$$i_{resi} = Depth2Index(d_{orig}) - Depth2Index(d_{pred}) \qquad (1)$$

where $d_{org}$ denotes the original depth value. $d_{pred}$ denotes the predicted depth value and $Depth2Index(.)$ denotes the Index lookup table. At the decoder side, the reconstructed depth value is derived by:

$$d_{rec} = Index2Depth(Depth2Index(d_{pred}) + i_{resi}) \qquad (2)$$

where $Index2Depth(.)$ denotes the depth value lookup table.

## 2.2. View synthesis optimization

In a 3DV system, depth and texture of the captured views are encoded together and the virtual views are generated with view synthesis from the coded views. In certain 3DV encoders, the ultimate goal of depth coding is not to improve the quality of the depth images themselves, but to improve the quality of the synthesized views. VSO is implemented at the encoder to achieve rate-distortion optimization coding of the depth. In VSO, when coding each prediction unit (PU) of a depth picture, the rate distortion (RD) cost of each mode of the PU is decided not only by the distortion between the reconstructed pixels and the original pixels; but also by the distortion evaluated at the synthesized views, i.e., synthesized view distortion change (SVDC) generated by comparing pixels synthesized by the reconstructed depth block and the pixels synthesized by the original depth block. Therefore, the RD cost is calculated as:

$$J = (i_{DWeight} \times D_{depth} + i_{VSOWeight} \times SVDC) / (i_{DWeight} + i_{VSOWeight}) + \lambda \times R \quad (3)$$

where $i_{VSOWeight}$ is the weighting factor. $SVDC$ is defined as the distortion difference $\Delta D$ between two synthesized textures $s'_T$ and $\tilde{s}_T$,

$$\Delta D = \tilde{D} - D = \sum_{(x,y)\in I} [\tilde{s}_T(x,y) - s'_{T,\mathrm{Ref}}(x,y)]^2 - \sum_{(x,y)\in I} [s'_T(x,y) - s'_{T,\mathrm{Ref}}(x,y)]^2 \quad (4)$$

$s'_{T,\mathrm{Ref}}(x,y)$ denotes a reference texture rendered from the 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 Equation (4) is also depicted in Figure 2[1]. VS denotes the view synthesis process and SSD stands for sum of squared differences. $s'_T$ denotes a texture rendered from a depth image $s_D$ 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 the original depth data as well. For the synthesis of the texture $\tilde{s}_T$ a depth image $\tilde{s}_D$ is used that differs from the depth image $s_D$ in that it contains the distorted depth data also for the current block $B$.

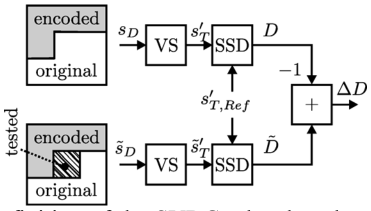


**Fig. 2.** Definition of the SVDC related to the distorted depth data of the block depicted by the hatched area in the bottom branch.

The above two techniques exploit the characteristics of depth images and can improve coding efficiency significantly. However, the existing depth processing methods didn't take advantages of these two techniques. To achieve better synthesis view and higher coding efficiency, in this paper, we propose the framework of sample edge offset compensation, which focuses on compressing the depth value and improving the edge areas and thus improve the synthesis view.

## 3. FRAMEWORK OF SAMPLE EDGE OFFSET COMPENSATION

A pixel in depth image corresponds to a disparity in synthesis view. More specifically, a pixel value corresponds to a horizontal displacement of the co-located pixel in the texture image when rendering the synthesis view. Therefore, the distorted depth image will result in distorted horizontal displacement. If the texture area corresponding to the current pixel is smooth in horizontal direction, the distortion in depth image may not cause large distortion in the synthesized view. Otherwise, if the pixel values in texture fluctuate significantly along horizontal direction, minor distortion of depth image may cause large distortion in the synthesized view. Preserving these characteristics and especially depth discontinuities is a crucial requirement for depth image coding. However, the distortion of compressed depth images may lead to ringing artifacts along the sharp edges. This may result in quality degradation when using these reconstructed depth images for view synthesis.

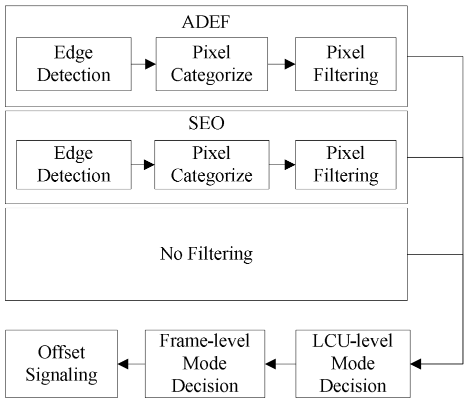


**Fig. 3.** The architecture of our proposed Sample Edge Compensation scheme.

In this section, we proposed a sample edge offset compensation algorithm for the reconstructed picture. Figure 3 shows the architecture of our proposed scheme. A two level mode decision is proposed for the selection of filtering and non-filtering, e.g. LCU-level and frame-level. At LCU-level, one of the multiple processing modes, including sample edge offset (SEO), adaptive depth edge filter (ADEF) and no-filtering, can be adaptively selected by calculating the rate distortion cost. To achieve a better coding performance, at frame-level, those RD costs are accumulated to decide whether the current frame needs to be processed by SEOC. If the total RD cost is less than zero, the filter is applied; otherwise we can skip the edge processing for all LCUs in this frame.

## 3.1. Adaptive depth edge filtering

For each pixel in a LCU, the smoothness is firstly checked by calculating the difference of the maximum and minimum depth intensity value within a neighborhood,

$$V_1 = |\max(I_W(i,j)) - \min(I_W(i,j))| \tag{5}$$

where $I_W(i,j)$ represents intensity values of the pixel $(i,j)$ in an n by n window (5x5 is used in our implementation) centered at the current pixel $(i,j)$. If $V_1$ is large enough, i.e., $V_1 > T_1$, the current pixel is regarded as an edge-like pixel and will be filtered in the next steps. In our implementation, $T_1$ is set to 10. Otherwise, the current pixel is considered as a smooth pixel in a homogeneous depth region, and will not be filtered.

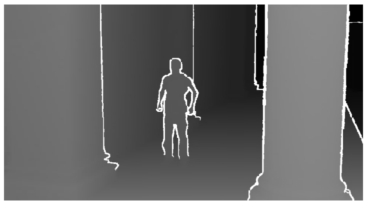


**Fig.4.** Pixels that are detected as edge-like pixels (marked with white color).

Figure 4 shows an example of the pixel locations that are detected as edge-like pixels. The pixels masked with white color are detected edge-like pixels. Those edge pixels are filtered as follows:

1 Calculate the mean depth value within 5x5 window by:

$$\bar{I_W} = \frac{1}{N^2}\sum_{(i,j)\in W} I(i,j) \tag{6}$$

2 Classify the pixels inside the window into two classes, $C_1$ and $C_2$, by:

$$C_1 = \{I(i,j)\}_{I(i,j)<\bar{I_W}}, C_2 = \{I(i,j)\}_{I(i,j)\geq\bar{I_W}} \tag{7}$$

3 Calculate the mean of each class:

$$\bar{I_{C1}} = \frac{1}{N_1}\sum_{(i,j)\in C_1} I(i,j), \bar{I_{C_2}} = \frac{1}{N_2}\sum_{(i,j)\in C_2} I(i,j) \tag{8}$$

4 For each class, find the representing pixel which has the minimum distance to the mean value in that class:

$$P_1 = \arg\min_{(i,j)\in C_1}\left|\bar{I_{C_1}} - I(i,j)\right|$$

$$P_2 = \arg\min_{(i,j)\in C_2}\left|\bar{I_{C_2}} - I(i,j)\right| \tag{9}$$

5 Set the filtered value as one of the values by:

$$\tilde{I}(P) = \begin{cases} I(P_1) & if\,|I(P)-I(P_1)| < |I(P)-I(P_2)| \\ I(P_2) & otherwise \end{cases} \tag{10}$$

i.e., the filtered value is the representing value of the class that has larger intensity similarity to pixel.

## 3.2. Sample edge offset

In this subsection, we'll introduce the proposed sample edge offset algorithm. Edge areas are firstly extracted and masked to be processed later. Then we categorize the masked pixels into multiple classes. For each class, an offset is added to the average of the reconstructed values to get the new reconstruction.

### *3.2.1. Edge detection*

As analyzed, different areas in texture have different tolerance to depth distortion. If the texture area corresponding to the depth pixel is smooth in horizontal direction, the distortion in depth image may not cause large distortion in the synthesized view. Otherwise, if the pixel values in texture fluctuate significantly along horizontal direction, minor distortion of depth image may cause large distortion in the synthesized view. That's to say only distortion around strong vertical edge will cause serious quality degradation on synthesized views. So we need to detect the edge areas firstly and then compensate accordingly. Here we apply the sobel operator to detect the vertical edge of each pixel of the depth block, which can be represented as:

$$E = d[y-1][x+1] + 2\times d[y][x+1] + d[y+1][x+1]$$
$$-(d[y-1][x-1] + 2\times d[y][x-1] + d[y+1][x-1]) \tag{11}$$

where $E$ represents edge strength. If the edge strength is larger than a threshold, the pixel is considered as an edge pixel. Moreover, the pixels within a given range with the edge pixel are also considered and masked for further processing.

### *3.2.2. Edge pixel categorization and Offset signaling*

Pixels belong to different patterns have different distortion. To better compensate pixels belong to different patterns. Before being filtered, the masked pixels are classified into four categories. Firstly they are classified into two categories according to their average value, the pixels with values smaller than the average value and the others larger than it. For each of the two categories, we further classified them into two categories. In total, these masked pixels are classified into four categories.

In pixel filtering, for each category, the average value of the original and the reconstructed depth pixels are calculated, and then the difference between them is identified and transmitted to the decoder side. For each category, an offset is used to compensate the reconstructed pixels which is calculated as:

$$Offset = d_{AvgOrg} - d_{Avg\,\mathrm{Rec}} \tag{12}$$

where $d_{AvgOrg}$ and $d_{Avg\,\mathrm{Rec}}$ represent the average value of original pixels and reconstructed pixels respectively. After filtering, the value of each class is replaced by:

$$d'_{\mathrm{Rec}} = d_{Avg\,\mathrm{Rec}} + Offset \tag{13}$$

However, the offset sometimes is very large especially with large QPs. It is better to quantize the offset before coded. But if the quantitative granularity is fine, the coding bits are still significant; vice versa, we'll still observe large distortion on post-processed pixels. To decrease the bit for offset coding, the depth look-up table (DLT) is used. Instead of transmitting original offset, the index difference of the original and the reconstructed depth value transmitted, which is calculated as:

$$OffsetIdx = Depth2Idx(d_{AvgOrg}) - Depth2Idx(d_{Avg\,\mathrm{Rec}}) \quad (14)$$

In the decoder side, the post-processed reconstructed depth value is:

$$d'_{\mathrm{Rec}} = Idx2Depth(Depth2Idx(d_{Avg\,\mathrm{Rec}}) + OffsetIdx) \quad (15)$$

where $Depth2Idx$ and $Idx2Depth$ are the DLT and inverse DLT. The utilization of depth look-up table can resolve the two problems mentioned above, and it can do offset quantization without causing any distortion.

## 3.3. RDO process

Rate distortion optimization is a common way to balance the coding bit rate and quality as shown in Equation 16.

$$J = D + lambda \times R \quad (16)$$

where $D$ is the distortion, and $R$ is the bitrate given distortion $D$. In the conventional RDO process, D is calculated as the distortion between original and reconstructed image. However, the proposed filter is implemented on depth image. Since it is the synthesized view but not the depth image to be viewed directly, it's not reasonable only consider the distortion on depth image. For the distortion part, the quality of synthesized views should be considered. To get the distortion on synthesized views, view synthesis must be done. So, we use equation 3 and 4 to calculate $J$. Here the definition for SVDC is changed as shown in Figure 5. "Filtered Pic" refers to the filtered picture; "Recon Pic" refers to the reconstructed picture before filtering.

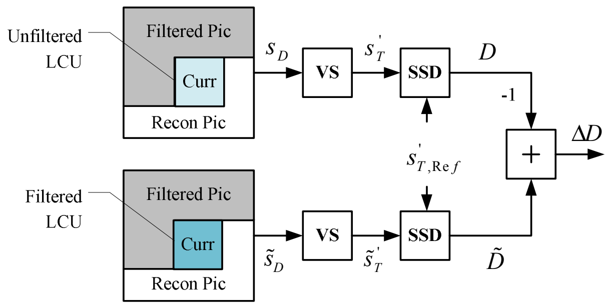


**Fig. 5.** Method of calculating distortion on synthesized views.

The mode with smaller J is chosen as the best mode for the current LCU. Then this RD cost is compared that without edge filtering. If the latter is smaller than the former, the LCU-level edge filtering is on and the best RD cost of the current LCU is that of the latter; vice versa. We add up all the best RD costs of LCU. If it is smaller than zero, then the frame-level edge filtering is on, vice versa.

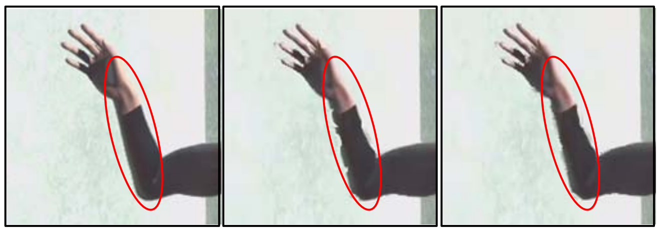
## 4. EXPERIMENTAL RESULTS

To validate the efficiency of the proposed algorithm, we integrate our scheme into the HTM7.0 reference software [18]. The common test conditions (CTC) are used [19]. Experiments were conducted on the common test sequences of 3 seconds. The coding performance is measured by BD-rate.

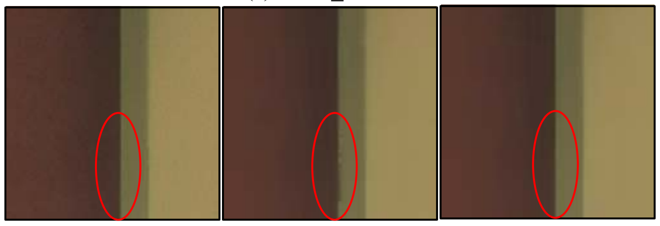
In Table 1, we tabulate the RD performance on both texture and synthesized view of the proposed algorithm. From the table, we can see that 2.03%-13.61% bitrate saving can be achieved on synthesized view by the proposed framework with minor coding gain on videos. And the average coding performance gain is 5.73%.

**Table 1.** Performance of the proposed scheme in HTM7.0.

| Sequences | video PSNR / video bitrate | synth PSNR / total bitrate |
|---|---|---|
| Balloons | -0.05% | -2.88% |
| Kendo | -0.09% | -2.14% |
| Newspaper_CC | 0.00% | -6.03% |
| GT_Fly | -0.09% | -2.03% |
| Poznan_Hall2 | 0.24% | -10.29% |
| Poznan_Street | -0.10% | -3.11% |
| Undo_Dancer | -0.16% | -13.61% |
| 1024x768 | -0.05% | -3.68% |
| 1920x1088 | -0.03% | -7.26% |
| **Average** | **-0.04%** | **-5.73%** |


Original HTM7.0 Proposed

(a) Undo_Dancer


Original HTM7.0 Proposed

(b) PoznanHall2

**Fig. 6.** Blocks clipped from the synthesized views.

In order to achieve an intuitive subjective visual experience of our algorithm, we clip a 176x176 block from synthesized views of Undo_Dancer and PoznanHall2 respectively. In Figure 6 (a) and (b), the blocks from three different sequences which are the synthesized views generated by the original depth images and the coded depth images with HTM7.0 and coded depth images with HTM7.0 plus proposed method are showed. The severely distorted areas are marked with red circle. In Figure 6(a), the original block is pretty smooth along the arm. However, in the block obtained from HTM7.0, we can see jagged distortion along the arm. With our proposed method, the smoothness is preserved and the visual performance has been largely improved. Although the distortion still exists, it is much smaller and hard to be perceived. From Figure 6(b), we can observe the same phenomenon. With the proposed framework, blurring artifacts around the pillar are removed.

The experimental results indicate that our proposed algorithm can efficiently improve the depth image in terms of both PSNR and visual quality of the synthesized views.

## 5. CONCLUSION

This paper proposes a sample edge offset compensation (SEOC) framework to improve the quality of depth images in a way that the quality of the synthesized views can be eventually improved. The novelty lies in that only areas that affect synthesized view quality are filtered and the filter selection is based on achieving the best balance in terms of bitrate and synthesized view quality. Experimental results show that both object and subject quality are improved with the proposed filtering framework.

## 6. REFERENCES

[1] G. Tech, K. Wegner, Y. Chen and S. Yea, "3D-HEVC Test Model 3", JCT3V-C1005_spec_d1, Geneva, 2013.

[2] A. Smolic, K. Müller, K. Dix, P. Merkle, P. Kauff, and T. Wiegand, "Intermediate view interpolation based on multiview video plus depth for advanced 3D video systems," in Proc. IEEE International Conference on Image Process. (ICIP), San Diego, CA, Oct. 2008.

[3] N. Zhang and S. Ma, "H.264/AVC-Based Depth Map Sequence Coding Using Improved Loop-filter," Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2009.

[4] J. He, S. Ma, N. Zhang and W. Gao, "Content adaptive in-loop depth map filter for HEVC based 3DV coding," IEEE International Conference on Speech and Signal Processing (ICASSP), 2013.

[5] V. Nguyen, D. Min and M. Do "Efficient edge-preserving interpolation and in-loop filters for depth map compression," 19th IEEE International Conference on Image Processing (ICIP), 2012.

[6] W. Hu, Au, O.C., L. Sun, W. Sun, L. Xu and Y. Li, "Adaptive depth map filter for blocking artifacts removal and edge preserving," IEEE International Symposium on Circuits and Systems (ISCAS), 2012.

[7] I. Lim, H. Wey and J. Lee, "Region-based adaptive bilateral filter in depth map coding," 18th IEEE International Conference on Image Processing (ICIP), 2011.

[8] F. Jager and J. Balle, "Median trilateral loop filter for depth map video coding," Picture Coding Symposium, 2012.

[9] S. Liu, P. Lai, D. Tian and C. Chen, "New Depth Coding Techniques With Utilization of Corresponding Video," IEEE Transactions on Broadcasting, vol.57, no.2, pp.551-561, June 2011.

[10] R. Zhong, R. Hu, Z. Wang, L. Liu and Z. Han, "LBP-Guided Depth Image Filter," Data Compression Conference (DCC), 2013.

[11] H. Ko and C.-C.J. Kuo, "A new in-loop filter for depth map coding in HEVC," Signal & Information Processing Association Annual Summit and Conference, 2012.

[12] S. Jung, "Enhancement of Image and Depth Map Using Adaptive Joint Trilateral Filter," IEEE Transactions on Circuits and Systems for Video Technology, vol.23, no.2, pp.258-269, Feb. 2013.

[13] F. Jager, M. Wien and P. Kosse, "Model-based intra coding for depth maps in 3D video using a depth lookup table," 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2012.

[14] G. Sullivan, J. Ohm, W. Han and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," IEEE Transactions on Circuits and Systems for Video Technology, vol.22, no.12, pp.1649-1668, 2012.

[15] G., S. H. Tech, K. Müller and T. Wiegand, "3D video coding using the synthesized view distortion change," Picture Coding Symposium (PCS), 2012 , vol., no., pp.25,28, 7-9 May 2012.

[16] K. Muller, P. Merkle, G. Tech and T. Wiegand, "3D video coding with depth modeling modes and view synthesis optimization," Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012.

[17] R. Zhang, Y. Chen and M. Karczewicz, "Adaptive Depth edge sharpening for 3D video depth coding," Visual Communications and Image Processing (VCIP), 2012.

[18] https://hevc.hhi.fraunhofer.de/svn/svn_3DVCSoftware /tags/HTM-7.0/.

[19] D. Rusanovskyy, K. Müller and A. Vetro, "Common Test Conditions of 3DV Core Experiments," JCT3V D1100, 4th Meeting: Incheon, KR, 20–26 April 2013.