Motion Compensation-based Low-Complexity Decoder Side Depth Estimation for MPEG Immersive Video

Patrick Garus, Felix Henry, Thomas Maugey, Christine Guillemot

Motion Compensation-based Low-Complexity Decoder Side Depth Estimation for MPEG Immersive Video

Patrick Garus
Orange Labs
Rennes, France
patrick1.garus@orange.com

Felix Henry
Orange Labs
Rennes, France
felix.henry@orange.com

Thomas Maugey
Inria
Rennes, France
thomas.maugey@inria.fr

Christine Guillemot
Inria
Rennes, France
christine.guillemot@inria.fr

***Abstract*—Decoder-Side Depth Estimation (DSDE) is a system firstly enabled in the novel MPEG Immersive Video (MIV) coding standard. In DSDE, only texture components are coded, while the depth is estimated at the decoder-side. This is motivated by previous work, which has shown high coding gain and pixel rate savings in DSDE. However, the computational complexity remains a concern, as high quality depth search has a high runtime and memory requirement. In this work we extend the concept of depth estimation to depth recovery. Using this mode, the decoder-side depth information is recovered through motion compensation utilizing the displacement vectors contained in the texture bitstream. This strategy enables us to replace most of the complex depth estimation processes with a simple motion compensation step, a decision that is drawn on the encoder-side and signaled per coding unit. With only minor losses in terms of synthesis PSNR and similar perceptual quality in terms of MS-SSIM, the complexity is significantly reduced. Depending on the acceptable loss, up to 80% of the moving objects depth may be motion compensated instead of estimated by a depth estimator translating into a speed-up of a factor of 104 for inter-frames compared to the reference depth estimator.**

***Index Terms*—Decoder Side Depth Estimation, MPEG Immersive Video, Immersive Video Coding**

I. Introduction

The Motion Pictures Experts Group (MPEG) finalized a novel immersive video standard denoted as MPEG Immersive Video (MIV) [1] [2]. It enables the user to freely navigate through captured or computer-generated volumetric video, *i.e.* enabling 6 Degrees of Freedom (DoF). It is compatible with the common multiview plus depth (MVD) format as well as with the emerging Multi-Plane Image format [3]. While it may be considered as a successor of 3D-HEVC, it does not follow the technical foundations of 3D-HEVC [4], namely inter-view and inter-component prediction, and specialized depth coding tools. Instead, the MIV is designed to operate on top of existing 2D video codecs and serves to describe the atlas format, which is the output of the MIV encoder. The atlas construction includes a pruning step which extracts images patches considered non-redundant among the views. This process is followed by the coding of depth and texture atlases independently using a 2D video codec. The decision to remain codec agnostic particularly impacts the coding of depth maps, as they cannot benefit from depth coding tools impacting the coding gain and perceptual quality of the rendering. Consequently, a Geometry Absent (GA) profile was introduced, which enables Decoder-Side Depth Estimation (DSDE) bypassing all issues related to the depth coding of MIV. In DSDE, the coding efficiency is significantly improved at the cost of increased client-side complexity.

In this paper, we propose a solution to the complexity problem of the GA profile, which is based on motion compensating the depth maps using the motion information provided by the texture decoder. This allows to omit the majority of depth estimation processes for Inter frames, which significantly reduces the runtime. All modern video codecs, such as AVC [5], HEVC [6] or VVC [7], will apply some form of motion compensation on the texture components. We re-use this available information to perform motion compensation on the depth component on a block-basis. This decision is drawn on the encoder-side and signaled. We avoid the transmission of residual information, as depth coding and modifications to the 2D codec is restricted by the MIV design. Consequently, depth estimation will be used only if the motion compensation fails given a certain error tolerance. Furthermore, this proposal presents an opportunity to think of depth estimation as part of the whole decoding framework by involving the texture decoder. Given the high potential in reducing the complexity of DSDE, we refer to our proposal as Low-Complexity DSDE (LC-DSDE).

The remainder of this paper is structured as follows: in section II, we introduce the MIV standard and the latest research related to DSDE, in section III we describe our proposed Low-Complexity DSDE approach, in section IV we discuss the results and section V concludes the paper.

II. Decoder Side Depth Estimation in MPEG Immersive Video

The choice and design of the atlas construction tools of MIV are mostly motivated by reducing pixel rate. Pixel rate is defined as the number of luma pixels that can be processed by a hardware video decoder. A test sequence with a resolution of 2048 × 1088 and 16 views requires 64 Megapixels at 30

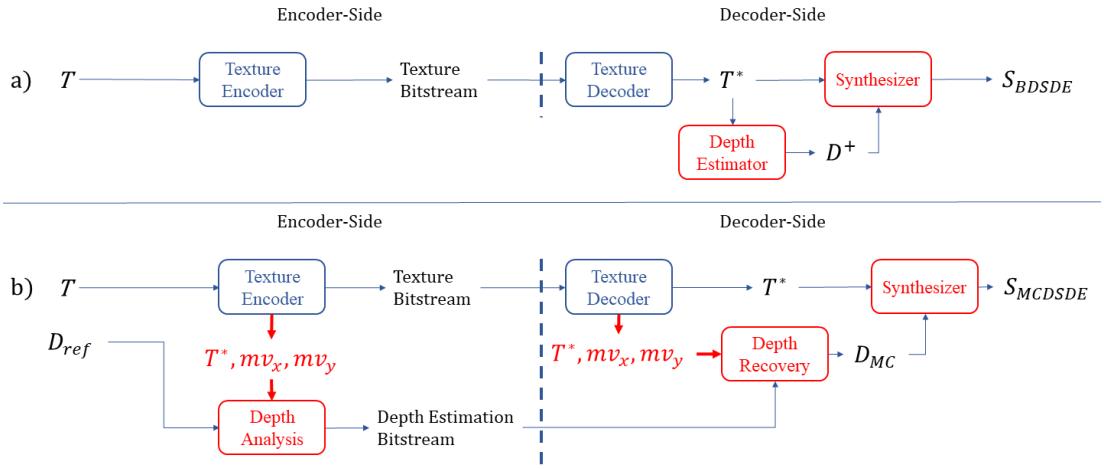

Fig. 1. Our proposed system. T and T^* are uncoded and coded textures. D_{ref} are reference depth maps. D_{MC} are depth maps, where some blocks have been recovered by motion compensation. D^+ are DSDE depth maps estimated by ffDE. Motion vectors are denoted as mv_x and mv_y. The resulting synthesized views are S_{BDSDE} for the Basic-DSDE system as well as S_{MCDSDE} for the DSDE system involving motion compensated depth.

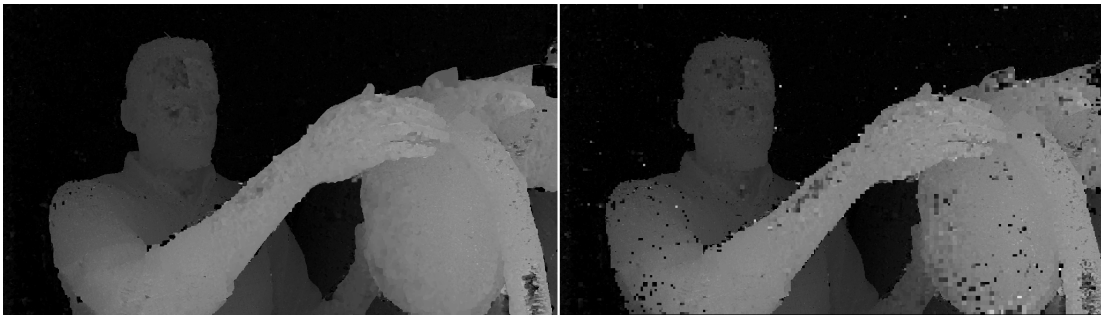

Fig. 2. Left: unmodified DERS8 depth (ffDE). Right: modified block based DERS8 depth (bbDE).

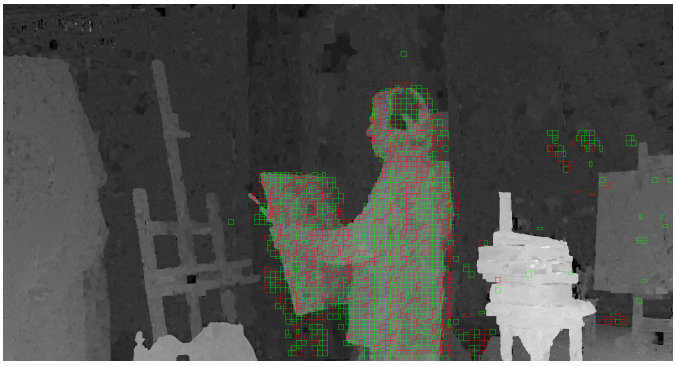

Fig. 3. Example for identified Prediction Units with non-zero motion information. Green blocks are motion compensated, while red blocks are estimated using bbDE (case $s = 0.25$, Painter sequence).

frames per second and therefore, only 8 out of 16 views can be coded into the atlases. Given that the CTC of MIV contains sequences with up to 64 views, the example clearly shows the demand for reducing pixel rate and why DSDE has become an attractive system.

The basic DSDE (B-DSDE) system is shown in Fig. Ia), which has been firstly evaluated in [12]. Textures T are coded using any 2D video encoder. At the decoder side, decoded textures T^* are used to estimate corresponding depth maps D^+. A synthesizer is then rendering intermediate views S_{BDSDE}. The DSDE system has been studied throughout the MIV standardization process and adopted as the Geometry Absent profile [19]. Given the high coding efficiency, the single concern of this system is the client-side complexity. In order to support the depth estimator in estimating depth maps faster and with higher accuracy, a dedicated Geometry Assistance SEI message has been adopted based on [18]. This approach allows to move significant amount of complexity to the encoder-side. However, more tools are necessary to further reduce the complexity of DSDE. One way is to exploit synergies with other modules of the DSDE system. Our proposal falls in this class and is therefore only possible in a coding system, taking advantage of the presence of the 2D texture decoder.

Beyond architectural and signaling related improvements, the depth estimator itself can be adapted to the context of DSDE. A point-to-block matching strategy improves the performance if depth are estimated from decoded textures [13]. Furthermore, a depth refinement process can be used to further improve the quality of depth maps estimated from decoded

textures [14].

III. Proposed Depth Recovery

It is known from 3D-HEVC studies that the motion in textures often correlate with the motion in the depth maps and therefore may serve as an useful predictor in the context of inter-component coding [20]. In LC-DSDE, we try to utilize the motion information already coded in the texture bitstream to motion compensate the depth maps and therefore, reconstruct depth for Inter frames, without using depth estimation. The system is shown in Fig. I. Again, we assume that the textures have been coded prior to the depth analysis on the encoder-side. This allows us to fetch the motion information from the texture bitstream and test it on the encoder-side for their accuracy of depth reconstruction. For this, we adopt the partitioning provided by the texture encoder, where a displacement vector is given for any prediction unit. Consequently, we can decide in the corresponding depth map, for any patch as small as a prediction unit, if it shall be compensated or estimated. In order to design such a decision, the block-based approach, presented in Chapter 5, has to be reconfigured to be applicable for very small, varying patches. Previous work has shown that such a strategy introduces artifacts in the corresponding depth [18]. Therefore, besides of the distortion introduces by motion compensation, the distortion introduced by a block-based approach has to be considered as well.

The depth recovery module on the decoder-side is utilizing the modified block-based depth estimator. The bitstream contains the result of the depth analysis and communicates to the depth recovery, which prediction units shall be compensated using the decoded depth at a previous time instance and which shall be computed using depth estimation. The first frame is estimated as usual with the original depth estimator, as Intra frames do not benefit from our approach. For subsequent Inter frames, the proposal is used.

After coding all textures, the depth analysis is performed. For all inter-coded CUs a displacement vector $[mv_x, mv_y]$ is fetched from the texture bitstream. The FFMPEG opensource library provides tools to extract these displacement vectors from the bitstream in a simple way for h.264/AVC [5]. As TMIV is compatible with any 2D codec the usage of AVC in the following experiments is possible. Furthermore, the partitioning associated with all displacement vectors is available. The displacement vectors, the decoded Textures T^* as well as the reference depth maps D_{ref} are provided to the Depth Analysis module. This module performs the following tests for all inter CUs [1]:

1) Perform block-based depth estimation on the CU and derive D_{bbDE}.
2) Perform motion compensation utilizing the previous intra-coded depth frame and the associated displacement vectors $[mv_x, mv_y]$ and derive D_{mc}.
3) Compute the L_2-distance between the reference depth D_{ref} and the two depth map candidates D_{mc} and D_{bbDE}.
4) Select the method based on the evaluation $L_{2,bbDE} > sL_{2,mc}$, with the error tolerance factor s. If the statement is true, the CU is recovered at the decoder-side through motion compensation. If the statement is false the CU is estimated by the block-based depth estimator.

The decision is signaled as part of the MIV bitstream and does not require normative changes to the 2D encoder.

[1]We continue the usage of *Coding Tree Unit* and *Coding Unit* to stay consistent with the text and the usage in more recent codecs like HEVC in VVC. In AVC, the CTU is denoted as a macroblock.

IV. Experimental Setup

We use five test sequences provided by the Common Test Conditions for MPEG Immersive Video [15], but differ in some aspects. In order to study the potential of our proposal, the configuration of the encoder has been drastically modified to disable intra coding in inter-frames. Furthermore, we disable B-Frames, as it will simplify the interpretation of the extracted motion information. In practice, the number of valid prediction units for motion compensation has to be expected to be smaller than in our experiments. On the other hand, the prediction is expected to be more accurate if B-Frames are enabled. Furthermore, more recent coding standards may provide displacement vectors with higher accuracy. Finally, we will investigate the high-bitrate scenario ($Q_T = 25$) for 5 sequences, excluding the still images of the CTC, as they do not benefit from this approach.

We utilize the depth estimation reference software (DERS8), developed and continuously improved by MPEG [8]. The configuration and settings are refined for the sequences of the CTC and we adopt them without modification. In order to utilize our proposal, DERS8 has to be adapted to support very small blocks, *i.e.* estimate depth information independently for each CTU. In AVC, the block size of motion compensated CUs can vary between 4×4 to 16×16. We enable DERS8 to extract the partitioning from the AVC bitstream and to estimate each motion compensated patch individually. It is therefore an extension to the block-based approach presented in [18]. Estimating depth for very small blocks naturally has an impact on the estimation quality. An example of the resulting depth quality is shown in Fig. 2. The left depth map originates from the unmodified full-frame DERS8 (ffDE), while the right depth map was generated using the block-based DERS8 (bbDE). The "blockiness" can be clearly seen, surprisingly however, the depth map appears globally correct.

The amount of motion compensated blocks is controlled by tolerating a higher error, *i.e.* we define a block as a motion compensated block if $L_{2,bbDE} > sL_{2,mc}$ with an error tolerance s and the L_2 distance to a reference depth D_{ref}. Here, we use the unmodified DERS8 depth maps as a reference. We test the values $s = [0.1, 0.25, 0.50, 0.75]$. An example of the chosen blocks is shown in Fig. 3 For each value s, we compute the amount of motion compensated blocks as well as the resulting synthesis PSNR and MS-SSIM [17]. Furthermore, the depth estimation runtime is provided comparing the block based DERS8 with different

TABLE I

AVERAGE SYNTHESIS PSNR AND MS-SSIM AS WELL AS THE AMOUNT OF MOTION COMPENSATED BLOCKS AND RUNTIME DIFFERENT THRESHOLDS S AND THE FFDE REFERENCE. THE PERCENTAGE REFERS ONLY TO BLOCKS WHICH HAVE AN IDENTIFIED MOTION.

Sequence	synth PSNR [dB]					synth MS-SSIM					Motion Compensated [%]				Depth Estimation runtime [s]				
	0.1	0.25	0.5	0.75	ffDE	0.1	0.25	0.5	0.75	ffDE	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	ffDE
Painter	33.89	34.23	34.36	34.39	34.88	0.968	0.972	0.973	0.973	0.976	88.71	77.37	60.32	47.32	2.7	8.1	15.2	22.3	412
Kitchen	31.69	31.84	31.89	31.90	32.02	0.975	0.976	0.977	0.977	0.977	56.41	42.48	29.19	20.29	1.9	3.6	5.9	6.5	594
Dancing	30.45	30.54	30.59	30.60	30.76	0.967	0.968	0.968	0.968	0.970	79.57	65.62	49.19	37.45	2.3	4.9	9.6	14.0	1467
Chef2	30.50	30.61	30.66	30.63	31.36	0.948	0.950	0.951	0.950	0.958	89.81	76.33	55.11	42.39	2.5	11.8	27.7	42.0	1485
Frog	26.27	26.96	27.17	27.21	27.90	0.913	0.927	0.930	0.930	0.940	76.65	51.97	31.09	21.70	35	98	149	170	657
Average	30.56	30.83	30.93	30.94	31.38	0.954	0.957	0.960	0.960	0.964	78.23	62.75	44.98	33.83	8.8	25.28	41.5	51.0	923

thresholds s and the unmodified DERS8. Computation was performed on an Intel Xeon (R) CPU E5-2520 @ 2.00 GHz. As the bitrate required to signal this decision is negligible in the immersive video context, we focus on the objective and subjective quality of the depth maps and the synthesizes views. The more blocks are motion compensated, the better the reduction in complexity. Simultaneously, more errors are introduced into the depth maps.

V. EXPERIMENTAL RESULTS

We first investigate the impact of our proposal on the depth maps quality, secondly on the view synthesis quality and finally investigate the benefit on the main goal of the proposal: the complexity reduction. Fig. 4 visualizes depth maps as well as corresponding synthesis for all thresholds as well as the ffDE reference. We show the last frame of the GOP, frame 16, which reflects the highest error. Furthermore, we chose to show patches containing motion, in order to make the artifacts of our proposal visible. In the following, we refer to Fig. 4 when discussing the visual performance.

A. Depth Maps Quality

In general, a quality degradation is expected in all cases, since no residual is allowed to be send, as otherwise the approach would steer towards depth coding. The impact of the last frame in the GoP is most severe for $s = 0.1$, also due to error propagation. However, it is apparent that even for $s = 0.75$ an increase in artifacts may appear. This is due to erroneously estimated depth in a block-based approach. As expected, most artifacts become visible if previously occluded objects appear because they cannot be recovered by motion compensation.

B. View Synthesis Quality

The degradation of view synthesis quality is quite severe for $s = 0.1$, with strong artifacts and visible "holes". Nevertheless, we observe that many artifacts are not due to the motion-compensation of depth, but rather due to the block-based approach of DERS8, because several artifacts appear at areas without any motion. As expected, most artifacts become visible if previously occluded objects appear because they cannot be recovered by motion compensation. If a high amount of error is tolerated ($s = 0.1$), the typical motion compensation artifacts in depth maps translate into an increased level of noise-like artifacts in the synthesized textures. Yet, even for $s = 0.1$ most of the synthesized objects are reconstructed accurately.

C. Complexity-Distortion Compromise

The main goal of the proposed LC-DSDE is to minimize the complexity (with minor impact on quality), as it is the main concern of DSDE. The objective metrics and the runtimes are shown in Tab. I for all tested sequences and thresholds s. The unmodified DERS8 is denoted as ffDE. We observe the following: first, the synthesis PSNR degrades with a lower threshold, and therefore, with more motion compensated blocks, as expected. Naturally, this is dependent on the amount of motion in sequence. Therefore, Frog has a loss of $1.63dB$, Painter and Chef2 losses of around $1dB$. In contrast, sequences with little motion have a lower degradation at $s = 0.1$. This is confirmed by the MS-SSIM metric. However, $s = 0.1$ is rather an extreme-case. Another interesting observation is, that the synthesis PSNR is rather similar between $s = 0.5$ and $s = 0.75$. This indicates that with $s = 0.5$, the majority of "safe to use" motion vectors are found, which do not significantly impact the quality in contrast to depth estimation. Again, this is confirmed by the MS-SSIM metric, which indicate the same value of 0.960 for $s = 0.5$ and $s = 0.75$. The next observation is, that despite the saturation of quality from $s = 0.5$, there is still a large gap towards the unmodified, full frame DERS8 synthesis PSNR. On average, a loss of $0.44dB$ is seen, with the smallest loss of $0.12dB$ for Kitchen and the largest loss of $0.73dB$ for Chef2. Due to the saturation of quality starting from $s = 0.5$, we conclude, that large parts of the degradation starting from this threshold are due to the block-based DERS8 approach and not due to the applied motion compensation.

Tab. I also summarizes the amount of motion compensated blocks, taking only into account the blocks with non-zero motion vectors. In case of $s = 0.1$, on average about 80% of blocks with indicated motion are used to motion compensate the depth blocks, instead of estimating them. The fraction gradually increases with the threshold to around 34% with $s = 0.75$. While we observed a saturation of quality loss from $s = 0.5$, there is indeed more blocks being compensated from $s = 0.75$ to $s = 0.5$. The runtime is also shown in this Table for all thresholds and compared to the unmodified DERS8. Naturally, the runtime of the unmodified DERS8 is much larger, as it is applied on the full frame. Nevertheless, focusing the depth estimation on blocks, which are not compensated, can provide a speed-up from 105 ($s = 0.1$) to 18 ($s = 0.75$). Considering the quality degradation, the optimal point among the tested thresholds is $s = 0.5$, for which a speed-up of 22.2

Fig. 4. Depth maps and synthesized view of four CTC sequences. Frame 16 is shown for the unmodified reference in the B-DSDE system using ffDE as well as the LC-DSDE proposal for different thresholds.

for inter-frames is measured.

VI. Conclusion

In this paper, LC-DSDE was presented, which takes advantage of motion information contained in the texture bitstream in order to motion compensate the depth information instead of estimating it. It was shown that a significant speed-up of factor 105 to 18 can be achieved for varying levels of tolerated error. This proposal is further a proof of concept for the potential of DSDE systems benefiting from jointly operating the texture decoder and the depth estimator.

References

[1] J. M. Boyce, R. Doré, A. Dziembowski, J. Fleureau, J. Jung, B. Kroon, B. Salahieh, V. K. M. Vadakital and L. Yu, "MPEG Immersive Video Coding Standard," Proc. of the IEEE.

[2] M. Wien, J.M. Boyce, T. Stockhammer, W.-H. Peng, "Standardization Status of Immersive Video Coding," IEEE J. on Emerging and Selected Topics in Circuits and Systems, vol. 9, no. 1, pp. 5-17, Mar. 2019.

[3] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N.K. Kalantari, R. Ramamoorthi, R. Ng and A. Kar,"Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines," ACM Transactions on Graphics (TOG), 2019.

[4] G. Tech, Y. Chen, K. Müller, J.-R. Ohm, A. Vetro, Y.-K. Wang, "Overview of the Multiview and 3D Extensions of High Efficiency Video Coding," IEEE Trans. on Circuits and Systems for Video Technology, Jan. 2019;26(1):35-49.

[5] D. Marpe, T. Wiegand and G. J. Sullivan, "The H.264/MPEG4 advanced video coding standard and its applications," in IEEE Communications Magazine, vol. 44, no. 8, pp. 134-143, Aug. 2006.

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

[7] M. Wien and B. Bross, "Versatile Video Coding – Algorithms and Specification," 2020 IEEE Int. Conf. on Visual Communications and Image Processing (VCIP), 2020.

[8] T. Senoh, N. Tetsutani, H. Yasuda and M. Teratani, "Revised Proposed Depth Estimation Reference Software (pDERS8.1)," ISO/IEC JTC1/SC29/WG11/MPEG2018/m45265.v3, Jan. 2019.

[9] M. Milovanović, F. Henry, M. Cagnazzo and J. Jung, "Patch Decoder-Side Depth Estimation in MPEG Immersive Video," IEEE Int. Conf. on Acoustics, Speech and Signal Process. (ICASSP), pp. 1945-1949, 2021.

[10] D. Mieloch, A. Dziembowski, J. Stankowski, O. Stankiewicz, M. Domański, Gwangsoon Lee, and Yun Young Jeong,"Immersive video depth estimation," ISO/IEC JTC1/SC29/WG11 m53407, Apr. 2020.

[11] D. Mieloch, O. Stankiewicz, and M. Domański,"Depth Map Estimation for Free-Viewpoint Television and Virtual Navigation," IEEE Access, vol. 8, pp. 5760–5776, 2020.

[12] P. Garus, J. Jung, T. Maugey and C. Guillemot, "Bypassing Depth Maps Transmission For Immersive Video Coding," Picture Coding Symposium (PCS), pp. 1-5, 2019.

[13] D. Mieloch, D. Klóska and M. Woźniak, "Point-to-Block Matching in Depth Estimation.," Int. Conf. on Computer Graphics, Visualization and Computer Vision 2021.

[14] D. Mieloch, A. Dziembowski and M. Domański, "Depth Map Refinement for Immersive Video," in IEEE Access, vol. 9, pp. 10778-10788, 2021.

[15] J. Jung, B. Kroon, and J. Boyce, "Common Test Conditions for Immersive Video," ISO/IEC JTC 1/SC 29/WG 11N18997, Feb. 2020.

[16] P. Boissonade and J. Jung, "[MPEG-I Visual] Improvement of VVS1.0.1," ISO/IEC JTC1/SC29/WG11/MPEG2019/m46263, Jan. 2019.

[17] Z. Wang, E. Simoncelli, and A. Bovik, "Multiscale structural similarity for image quality assessment," 37th Asilomar Conf. on Signals, Systems & Computers, vol. 2, pp. 1398–1402, 2003.

[18] P. Garus, F. Henry, J. Jung, T. Maugey and C. Guillemot, "Immersive Video Coding: Should Geometry Information Be Transmitted as Depth Maps?," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 32, no. 5, pp. 3250-3264, May 2022, doi: 10.1109/TCSVT.2021.3100006.

[19] D. Mieloch et al., "Overview and Efficiency of Decoder-Side Depth Estimation in MPEG Immersive Video," in IEEE Transactions on Circuits and Systems for Video Technology, doi: 10.1109/TCSVT.2022.3162916.

[20] Y.-L. Chan, C. Fu, H. Chen, and S.-H. Tsang, "Overview of current development in depth map coding of 3d video and its future", IET Signal Process., vol. 14, pp. 1–14, 2020.