Real-time CPU-based View Synthesis for Omnidirectional Video

Jakub Stankowski
Institute of Multimedia Telecommunications
Poznań University of Technology
Polanka 3
61-131 Poznań, Poland
jakub.stankowski@put.poznan.pl

Adrian Dziembowski
Institute of Multimedia Telecommunications
Poznań University of Technology
Polanka 3
61-131 Poznań, Poland
adrian.dziembowski@put.poznan.pl

ABSTRACT

In this paper, the authors describe the real-time CPU-based implementation of the virtual view synthesis algorithm for high-resolution omnidirectional content. The proposed method allows a user of the immersive video virtually navigating within the scene captured by a multiview system comprised of 360-degree or 180-degree cameras. The proposed method does not require using powerful graphic cards as other state-of-the-art real-time synthesis methods. Instead, the emerge of consumer-grade multithreaded CPUs and CPU-based virtual view synthesis, allows further development of cheap, consumer immersive video systems. The proposed method was compared with the state-of-the-art view synthesis algorithm – RVS, both in terms of quality of synthesized views and computational time required for the synthesis, presenting the usefulness of the proposed method.

Keywords

Virtual view synthesis, omnidirectional video, immersive video systems, real-time video processing.

1. INTRODUCTION

The virtual view synthesis is a crucial step of processing of immersive video [Isg14], virtual navigation of free-viewpoint television systems [Tan12]. It allows a user of such kind of the video system to virtually navigate within scene captured with a multicamera system (e.g., [Goo12], [Sta18], [Zit04]) by producing artificial viewports between actual positions of the cameras [Dzi19], [Fac18].

In order to provide a proper feeling of immersion of the user into the scene, the view synthesis has to be performed in the real-time.

There are multiple state-of-the-art methods of the real-time view synthesis. However, most of them require using dedicated FPGA devices (e.g. [Aki15], [Li19]), powerful GPUs (e.g., [Non18], [Zha17]) or even VLSI devices [Hua19]. The necessity of using such devices significantly limits the possibility of developing a cheap, consumer immersive video system.

In this paper, we present a real-time synthesis method implemented on the CPU, which is much harder to efficiently implement [Dzi18], [Sta20]. Moreover, existing algorithms of the CPU-based real-time view synthesis described in [Dzi18] and [Sta20] handle only the typical, perspective views thus cannot be used in modern immersive video systems with omnidirectional, 360-degrees video.

2. VIRTUAL VIEW SYNTHESIS FOR OMNIDIRECTIONAL CONTENT

Omnidirectional vs. perspective synthesis

Regardless of the content type, the virtual view synthesis is based on reprojecting information from input views to the virtual view. However, the math behind the reprojection differs for different types of content.

For perspective content, the reprojection uses homography matrices, combining extrinsic and intrinsic parameters of input camera and virtual camera [Sta20].

For omnidirectional content, reprojection equations depend on the representation type, e.g., equirectangular projection (ERP) or cube map projection (CMP). In this paper, we deal with the ERP, as it is the most commonly used representation for omnidirectional video.

Algorithm of view synthesis

The proposed algorithm is made up from four major stages (Fig. 1): (1) depth reprojection, (2) depth merging and filtering of the depth map of the virtual view, (3) texture reprojection, and (4) inpainting of holes in the virtual view.

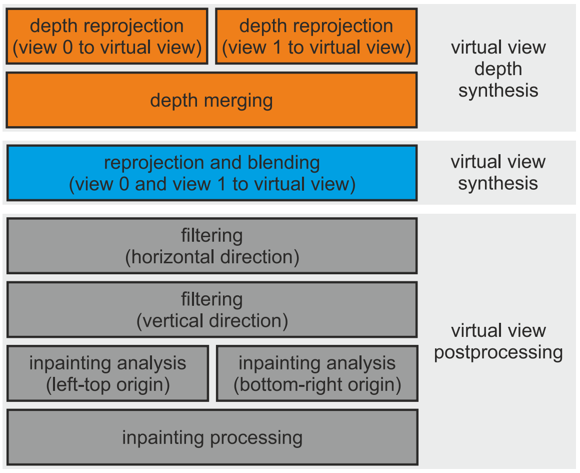
depth reprojection (view 0 to virtual view)
depth reprojection (view 1 to virtual view)
depth merging
virtual view depth synthesis
reprojection and blending (view 0 and view 1 to virtual view)
virtual view synthesis
filtering (horizontal direction)
filtering (vertical direction)
inpainting analysis (left-top origin)
inpainting analysis (bottom-right origin)
inpainting processing
virtual view postprocessing

Figure 1. Overview of described synthesis algorithm.

In order to meet the requirement of the computational time and make the algorithm real-time, only two input views are used for the synthesis. However, in the omnidirectional scenario, where each input view contains much more information than for perspective cameras, such a limitation does not significantly reduce the quality of the synthesized views.

Of course, in case of using only two views, these two views have to be carefully chosen among all available input views in order to provide the best possible quality. However, fast and efficient view selection algorithms exist (e.g. [Dzi18b] or the method used in [MPEG21b]) and can be used for this purpose.

2.1.1 Depth reprojection

At first, only depth maps are processed. Each pixel of both input depth maps is processed in the same way. Firstly, the position of the point in the 3D space is calculated:

$$\begin{aligned} X &= z_R \cdot cos(\theta_R) \cdot cos(\varphi_R) - t_X^R\,, \\ Y &= z_R \cdot sin(\theta_R) - t_Y^R\,, \\ Z &= -z_R \cdot cos(\theta_R) \cdot sin(\varphi_R) - t_Z^R\,, \end{aligned}$$

where z_R is the depth of the pixel, $[t_X^R, t_Y^R, t_Z^R]'$ is the translation vector of the input camera R, and (θ_R, φ_R) is the angular position of the pixel within input view R.

In the second step of depth reprojection, a 2D position of the pixel within virtual view (θ_R, φ_R) and its depth z_V are calculated:

$$\varphi_V = tan^{-1}\left(-\frac{Z - t_Z^V}{X - t_X^V}\right),$$

$$z_V = \sqrt{(X - t_X^V)^2 + (Y - t_Y^V)^2 + (Z - t_Z^V)^2}\,,$$

$$\theta_V = sin^{-1}\left(\frac{Y - t_Y^V}{z_V}\right).$$

If the virtual view is a perspective one, the projection from 3D space into the image plane is performed with multiplication by projection matrix of the virtual camera:

$$\begin{bmatrix} x_V \\ y_V \\ z_V \end{bmatrix} = \boldsymbol{P_V} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}.$$

2.1.2 Depth merging and filtering

In the second stage, depth maps reprojected from both input depth maps are merged to create a single depth map of the virtual view.

For each pixel of this depth map, the merging algorithm chooses the depth candidate, which is closer to the virtual camera. If a pixel was reprojected from one input view only, no merging is necessary, and this single candidate is chosen. If no depth candidates are available (i.e., pixel was not visible in any input view), the pixel of the merged depth map will be empty.

After merging, the depth map of the virtual view is being filtered to eliminate small depth artifacts visible as single pixels surrounded by much smaller or much greater depth values. Such wrong pixels in the depth map may significantly reduce both the subjective and objective quality of the synthesized view, and are caused mostly by object discontinuities and blurred edges within input depth maps.

2.1.3 Texture reprojection

After creation of the depth map of the virtual view, texture information is reprojected. The fast reprojection of texture is performed using look-up tables, which for each pixel of the virtual view store its initial position in the input view.

The color of each pixel is calculated by averaging of its colors in both input views (if possible) or copying a color from one input view if it was not visible in another view.

2.1.4 Inpainting

After the depth and texture reprojection, the virtual view has holes – areas not visible in any input views. These areas have to be inpainted [Ber00], [Dar10].

In order to perform possibly fastest inpainting, a color of each empty pixel is set by copying of the color from one of its four neighbors (top, left, right or bottom) – the neighbor which has the farthest depth.

3. ALGORITHM IMPLEMENTATION

In order to evaluate the synthesis performance several implementations have been prepared. The basic one described as "Reference" is written without any optimizations. The second one is algorithmically optimized including reduction of time-consuming operations, buffering of pre-calculated immediate values and memory load/stores reduction.

The second set of implementations includes vectorization and parallelization by using techniques developed for previously described synthesis algorithm for perspective video [Sta20].

The vectorized implementations use AVX2 or AVX512 extensions [Dem13]. Unfortunately, due to significantly higher number of calculations and usage of more complex functions (including many trigonometric transformations) the vectorized implementation for omnidirectional view synthesis is much more difficult. Both vectorized implementation uses a specially crafted routines for $sqrt$, tan^{-1} and sin^{-1} calculation. The implementations used to calculate abovementioned functions prioritize performance over precision and can be considered as seasonable compromise between speed and distortion introduced by computation errors.

The multithreaded implementation uses previously developed Independent Projection Targets (IPT) [Sta20] in order parallelize the depth reprojection. Depth reprojection is the most compute heavy step of synthesis algorithm and its parallelization allows to significant reduction of computation time.

4. EXPERIMENTS

Test sequences

The proposed view synthesis algorithm was tested on a test set containing four miscellaneous omnidirectional test sequences (Fig. 2):

1. ClassroomVideo, 4K×2K resolution, 16 full-360° cameras [Kro18],
2. Chess, 2K×2K resolution, 10 semispherical cameras placed on the sphere [Ilo19],
3. Cyberpunk, 2K×2K resolution, 10 parallel semispherical cameras [Jeo22],
4. Hijack, 4K×2K resolution, 10 parallel cameras with angle of view 180°×90° [Dor18].

The sequences are commonly used in immersive video applications, e.g., within ISO/IEC JTC1/SC29/WG04 MPEG Video Coding group [MPEG21].

Experiment setup

In the experiment, 9 implementations of proposed virtual view synthesis method were evaluated. The results are compared with the state-of-the-art method for omnidirectional view synthesis: RVS (Reference View Synthesizer) [Fac18], commonly used by individual researchers and the ISO/IEC MPEG group [MPEG18].

The implementations differ in usage of AVX2 and AVX512 instruction sets, multi-threading (MT), and Independent Projection Targets (IPT) technique which allows to use separate buffers for each thread during depth projection.

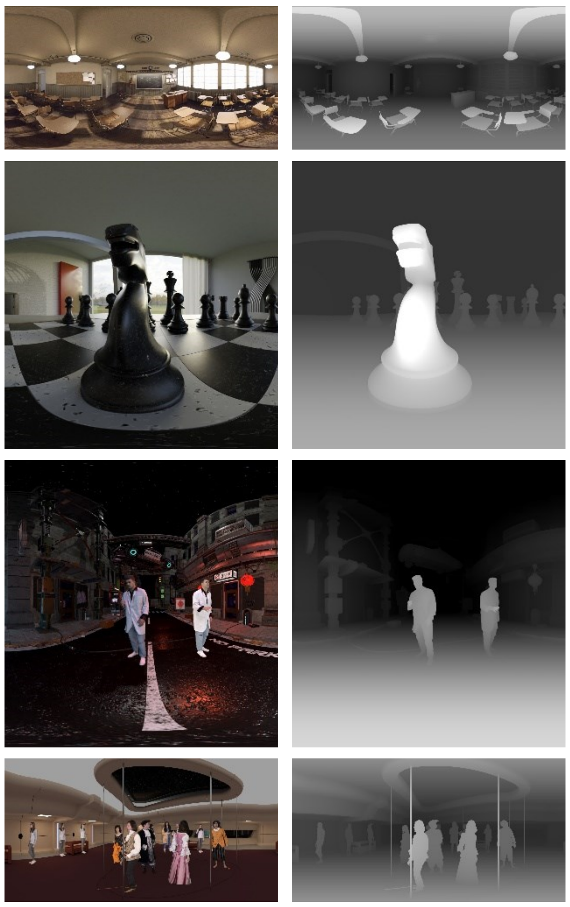
Figure 2. Input views and corresponding depth maps for (from top): ClassroomVideo, Chess, Cyberpunk, and Hijack.

CPU model	Implementation	Processing time [ms/frame]				
		DP	DM	VP	PP	Entire frame
i7-8700K	RVS	n/a	n/a	n/a	n/a	21583.300
	Reference	778.356	1.869	72.967	70.264	923.458
	Optimized	736.679	1.864	73.216	74.240	886.000
	Optimized + MT	375.859	1.889	12.184	14.664	404.597
	Optimized + MT + IPT	197.117	11.830	12.054	14.571	235.573
	Optimized + AVX2	116.948	1.846	50.350	73.066	242.210
	Optimized + AVX2 + MT	62.580	1.908	11.313	14.626	90.428
	Optimized + AVX2 + MT + IPT	38.271	13.289	11.246	15.395	78.203
R9-3900X	RVS	n/a	n/a	n/a	n/a	19718.100
	Optimized	659.097	2.145	84.709	69.443	815.396
	Optimized + MT + IRT	183.148	12.162	11.764	12.162	219.238
	Optimized + AVX2	146.462	2.239	62.278	68.305	279.284
	Optimized + AVX2 + MT + IPT	44.242	13.849	11.360	12.644	82.097
i9-7900X	Optimized + MT + IRT	233.397	6.940	13.021	11.587	264.946
	Optimized + AVX2	144.286	2.957	65.467	80.655	293.366
	Optimized + AVX2 + MT + IPT	52.648	6.907	9.187	12.392	81.135
	Optimized + AVX512	103.010	3.081	67.077	83.265	256.434
	Optimized + AVX512 + MT + IPT	36.605	6.812	9.232	9.865	62.515
	Optimized + AVX512 + MT + IPT, synthesis 4K -> 2K	37.547	1.727	3.892	4.360	47.527

Table 1. Computation time comparison of the state-of-the-art view synthesis method RVS [Fac18] and all tested implementations of proposed synthesis method on 4K×2K sequence (ClassroomVideo) Processing stages: DP – depth projection, DM – depth merging, VP – view projection, PP – postprocessing.

CPU model	Implementation	Processing time [ms/frame]				
		DP	DM	VP	PP	Entire frame
i7-8700K	RVS	n/a	n/a	n/a	n/a	11383.700
	Reference	424.173	0.849	34.664	38.501	498.187
	Optimized	379.340	0.804	32.197	39.616	451.957
	Optimized + MT	141.300	0.844	6.739	7.160	156.043
	Optimized + MT + IPT	85.406	5.365	5.965	7.976	104.712
	Optimized + AVX2	47.482	0.947	21.619	38.537	108.585
	Optimized + AVX2 + MT	27.340	0.864	5.859	6.844	40.907
	Optimized + AVX2 + MT + IPT	17.201	5.053	4.700	6.970	33.924
R9-3900X	RVS	n/a	n/a	n/a	n/a	9476.200
	Optimized	248.529	0.977	39.807	28.696	318.009
	Optimized + MT + IRT	71.964	6.039	4.979	4.871	87.853
	Optimized + AVX2	67.775	1.030	29.170	36.822	134.797
	Optimized + AVX2 + MT + IPT	16.752	6.890	5.526	6.613	35.781
i9-7900X	Optimized + MT + IRT	101.038	3.417	7.151	4.304	115.91
	Optimized + AVX2	59.672	1.223	33.816	39.869	134.58
	Optimized + AVX2 + MT + IPT	21.937	2.638	3.495	6.471	34.541
	Optimized + AVX512	46.590	1.432	35.491	35.875	119.388
	Optimized + AVX512 + MT + IPT	16.349	3.063	4.844	3.800	28.056

Table 2. Computation time comparison of the state-of-the-art view synthesis method RVS [Fac18] and all tested implementations of proposed synthesis method on 2K×2K sequence (Cyberpunk). Processing stages: DP – depth projection, DM – depth merging, VP – view projection, PP – postprocessing.

Quality metric	ClassroomVideo		Hijack		Cyberpunk		Average		
	RVS	Proposed	RVS	Proposed	RVS	Proposed	RVS	Proposed	Difference
WS-PSNR [Sun17]	31.76	31.53	38.36	38.17	28.64	28.76	32.92	32.82	-0.10
IV-PSNR [MPEG20]	44.79	44.23	46.01	46.33	37.47	37.57	42.76	42.71	-0.04
VMAF [Li16]	38.32	40.27	71.18	67.30	39.73	37.49	49.74	48.35	-1.38
SSIM [Wan04]	0.927	0.921	0.987	0.986	0.861	0.869	0.925	0.925	0.001
MS-SSIM [Wan03]	0.705	0.656	0.947	0.944	0.658	0.673	0.770	0.758	-0.012
VIF [She06]	0.366	0.346	0.793	0.773	0.341	0.326	0.500	0.482	-0.018

Table 3. Average quality of synthesized virtual views.

In order to present the efficiency of the proposed view synthesis algorithm, the computational time needed for view synthesis was evaluated on three different CPUs: Intel i7-8700K, AMD R9-3900X and Intel i9-7900X. Processors used for evaluation differs both in architecture and in number of available cores. The i9-7900X is the only one being capable of executing AVX512 instructions which were available for performance evaluation.

The complexity of each tested implementation was evaluated as an average processing time needed for synthesis of one frame of a virtual view. The processing time was measured using precision time stamps according to [MDNL20]. For implementations developed by paper authors the processing times for each processing stage (depth projection, depth merging, view projection and postprocessing) was also gathered.

The quality of synthesized views was assessed using 6 state-of-the-art full-reference objective quality metrics: Weighted-to-Spherically-Uniform PSNR (WS-PSNR) [Sun17], Structural Similarity Index Measure (SSIM) [Wan04], Multi-Scale SSIM (MS-SSIM) [Wan03], Visual Information Fidelity (VIF) [She06], Video Multimethod Assessment Fusion (VMAF) [Li16], and ISO/IEC MPEG's metric for immersive video – IV-PSNR [MPEG20].

The quality of the synthesis was estimated by comparing input views with virtual views synthesized at the same position.

Evaluation results

The results of performed experiments are presented in Tables 1 – 3. Tables 1 and 2 show the average computational time required for synthesis of one frame of the virtual view, for 4K×2K and 2K×2K sequence, respectively.

The performance of proposed synthesis technique has been measured as average computation time required to synthesize one video frame. Independently of the platform, even the unoptimized implementation of proposed technique was at least order of magnitude faster than RVS. The algorithmic and implementation related optimizations allows for ~5% reduction in computational time. The higher gain could be achieved by using AVX2 vectorized implementation (up to 87%). The change from AVX2 to AVX512 leads only to small improvements since the used processor (i9-7900X) combines two 256-bit execution units into one 512-bit. The most gain in AVX512 implementation comes from more efficient EVEX encoding, reduced processor front-end burden and usage of mask registers.

The parallelization techniques allow for significant improvements in synthesis performance but is strongly correlated with number of available CPU cores. The combined gain from parallelization techniques (typical multithreaded implementation + IPT) allows to speedup computations by 4 times.

Fortunately, both vectorization and parallelization can be constructively combined leading to almost 14× better performance when compared to optimized implementation.

For the 4K×2K test sequence the best measured performance is ~16 FPS (with ~21 FPS with reduced output resolution). This cannot be treated as real-time, but the value is close to 25 FPS and further improvements in CPU performance and some tuning of implementation could allow for real time processing.

For 2K×2K resolution the framerate of ~38 FPS was achieved implying, that the proposed virtual view synthesis algorithm can operate in the real-time for high-resolution immersive content.

In Table 3, average objective quality metrics for each sequence are presented. It has to be noted, that the quality of the virtual view does not depend on the implementation of the proposed synthesis method.

Fig. 3 presents the subjective comparison between fragments of views synthesized using RVS (left) and proposed method (right). The characteristics of synthesis artifacts are different because of different inpainting methods and the general rule of reprojection (triangle-based projection in RVS and fast pixel-based projection in the proposed algorithm). However, it can be stated that the overall subjective quality of views synthesized using both tested methods is similar.

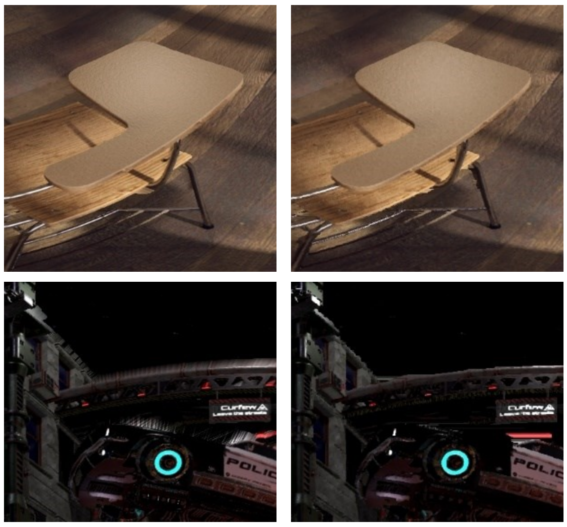
Figure 3. Fragments of virtual views synthesized using RVS (left) and proposed method (right).

5. CONCLUSIONS

The virtual view synthesis for omnidirectional views requires more calculations and is less susceptible to reprojection simplifications than for typical, perspective views. However, the paper shows, that the development of the CPU-based implementation of the real-time virtual view synthesis method is possible also for such kind of content.

The experimental results show that good-quality virtual views can be synthesized in the real-time, providing the possibility of development of cheap immersive video systems in the near future.

6. ACKNOWLEDGMENTS

This work was supported by the Ministry of Education and Science of Republic of Poland.

7. REFERENCES

[Aki15] Akin, A., Capoccia, R., Narinx, J., Masur, J., Schmid, A., and Leblebici, Y. Real-time free viewpoint synthesis using three-camera disparity estimation hardware. 2015 IEEE International Symposium on Circuits and Systems (ISCAS), Lisbon, pp. 2525-2528, 2015.

[Ber00] Bertalmio, M., Sapiro, G., Caselles, V., and Ballester, C. Image inpainting. SIGGRAPH 2000, New Orlean, USA, 2000.

[Dar10] Daribo, I., and Pesquet-Popescu, B. Depth-aided image inpainting for novel view synthesis. 2010 IEEE International Workshop on Multimedia Signal Processing, Saint Malo, France, 2010.

[Dor18] Doré, R. Technicolor 3DoF+ test materials. ISO/IEC JTC1/SC29/WG11 MPEG, M42349, San Diego, CA, USA, 04.2018.

[Dem13] Demikhovsky, E. Intel® AVX-512 Architecture. Comprehensive vector extension for HPC and enterprise, LLVM Developers' Meeting, San Francisco, USA, 2013.

[Dzi18] Dziembowski, A., and Stankowski, J. Real-time CPU-based virtual view synthesis. 2018 International Conference on Signals and Electronic Systems (ICSES), Kraków, Poland, 2018.

[Dzi18b] Dziembowski, A., Samelak, J., Domański, M., “View selection for virtual view synthesis in free navigation systems,” International Conference on Signals and Electronic Systems, ICSES 2018, Kraków, Poland, 10-12.09.2018.

[Dzi19] Dziembowski, A., Mieloch, D., Stankiewicz, O., Domański, M., Lee, G., and Seo, J. Virtual view synthesis for 3DoF+ video. 2019 Picture Coding Symposium (PCS), Ningbo, China, 2019.

[Fac18] Fachada, S., Bonatto, D., Schenkel, A., and Lafruit, G. Depth image based view synthesis with multiple reference views for virtual reality. 3DTV-Conference: The True Vision – Capture, Transmission and Display of 3D Video (3DTV-CON), Helsinki, Finland, 2018.

[Goo12] Goorts, P., Dumont, M., Rogmans, S., and Bekaert, P. An end-to-end system for free viewpoint video for smooth camera transitions. 2012 International Conference on 3D Imaging (IC3D). Liege, Belgium, 2012.

[Hua19] Huang, H., Wang, Y., Chen, W., Lin, P. and Huang, C. System and VLSI implementation of phase-based view synthesis. 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, United Kingdom, pp. 1428-1432, 2019.

[Ilo19] Ilola, L., Vadakital, V.K.M., Roimela, K., and Keraenen, J. New test content for immersive video – Nokia Chess. ISO/IEC JTC1/SC29/WG11 MPEG, M50787, Geneva, Switzerland, 10.2019.

[Isg14] F. Isgro et al. Three-dimensional image processing in the future of immersive media. IEEE Tr. on Circuits and Systems for Video Tech., 2014.

[Jeo22] Jeong, J.Y., Yun, K.J., Lee, G., Cheong, W.S., Yoo, S. "[MIV] ERP Content Proposal for MIV ver.1 Verification Test," ISO/IEC JTC1/SC29/WG04 MPEG VC, M58433, Online, Jan. 2022.

[Kro18] Kroon, B. 3DoF+ test sequence ClassroomVideo. ISO/IEC JTC1/SC29/WG11 MPEG, M42415, San Diego, CA, USA, 04.2018.

[Li16] Li, Z., Aaron, A., Katsavounidis, I., Moorthy, A., and Manohara, M. Toward a practical perceptual video quality metric. Netflix Technology Blog, 2016.

[Li19] Li, Y., Claesen, L., Huang, K., and Zhao, M. A real-time high-quality complete system for depth image-based rendering on FPGA. IEEE Transactions on Circuits and Systems for Video Technology, vol. 29, no. 4, pp. 1179-1193, 2019.

[MDNL20] Microsoft Developer Network Library. Acquiring high-resolution time stamps. https://msdn.microsoft.com/enus/library/windows/desktop/dn553408, 2020.

[MPEG18] "Reference View Synthesizer (RVS) manual," Doc. ISO/IEC JTC1/SC29/WG11 MPEG, N18068, Macao, Oct. 2018.

[MPEG20] Software manual of IV-PSNR for Immersive Video. ISO/IEC JTC1/SC29/WG04 MPEG VC, N0013, Online, Oct. 2020.

[MPEG21] Common Test Conditions for MPEG Immersive Video. ISO/IEC JTC1/SC29/WG04 MPEG VC, N0051, Online, Jan. 2021.

[MPEG21b] Test Model 11 for MPEG Immersive video. Document ISO/IEC JTC1/SC29/WG04 MPEG VC, N0142, Online, Oct. 2021.

[Non18] Nonaka, K., Watanabe, R., Chen, J., Sabirin, H., and Naito, S. Fast plane-based free-viewpoint synthesis for real-time live streaming. 2018 IEEE Visual Communications and Image Processing (VCIP), Taichung, Taiwan, pp. 1-4, 2018.

[She06] Sheikh, H.R., and Bovik, A.C. Image information and visual quality. IEEE Transactions on Image Processing, vol. 15, no. 2, pp. 430-444, 2006.

[Sta18] Stankiewicz, O., Domański, M., Dziembowski, A., Grzelka, A., Mieloch, D., Samelak, and J. A Free-viewpoint Television system for horizontal virtual navigation. IEEE Transactions on Multimedia, vol. 20, no. 8, pp. 2182-2195, 2018.

[Sta20] Stankowski, J., and Dziembowski, A. Fast view synthesis for immersive video systems. Proceedings of the 28. International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, WSCG'2020, Plzen, Czech Republic, 05.2020.

[Sun17] Sun, Y., Lu, A., and Yu, L. Weighted-to-Spherically-Uniform Quality Evaluation for Omnidirectional Video. IEEE Signal Processing Letters 24.9(2017):1408-1412.

[Tan12] Tanimoto M. et al. FTV for 3-D Spatial Communication. 2012 Proceedings of the IEEE, vol. 100, no. 4, pp. 905-917, 2012.

[Wan03] Wang, Z., Simoncelli, E.P., and Bovik, A.C. Multiscale structural similarity for image quality assessment. The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, vol. 2, pp. 1398-1402, 2003.

[Wan04] Wang, Z., Bovik, A.C., Sheikh, H.R., and Simoncelli, E.P. "Image quality assessment: From error measurement to structural similarity," IEEE Transactions on Image Processing, vol. 13, Jan. 2004.

[Zit04] Zitnick, C.L., Kang, S.B., Uyttendaele, M., Winder, S., and Szeliski, R. High-quality video view interpolation using a layered representation. ACM Transactions on Graphics, vol. 3, pp. 600-608, 2004.

[Zha17] Zhang, L., Li, Y., Zhu, Q., and Li, M. Generating virtual images for multi-view video. Chinese Journal of Electronics, vol. 26, no. 4, pp. 810-813, 2017.