Fast View Synthesis for Immersive Video Systems

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
Immersive video has become a popular research topic recently. However, there are no fast immersive video processing methods, which could be used in practical immersive video systems. In this paper the real-time CPU-based virtual view synthesis method is presented. The proposed method allows a viewer to freely navigate within acquired scene without necessity of using dedicated FPGA devices or powerful graphic cards. Presented view synthesis method can be used in practical immersive video systems, even for ultra-high resolution sequences. In order to present usefulness of proposed method, several implementations and use cases are discussed in the paper.

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

1. INTRODUCTION
In this paper we deal with the virtual view synthesis for immersive video systems. Such kind of systems allow a viewer to immerse into a scene, i.e. to virtually navigate within a scene that was captured by a set of arbitrarily located cameras [Goo12][Sta18][Zit04] (Fig. 1).

In order to provide the possibility of smooth navigation of a user, his or her viewpoint cannot be limited only to images captured by multiple cameras – a user should be able to watch the scene from any, arbitrarily chosen position (orange camera in Fig. 1). In order to generate additional images, the virtual view synthesis operation should be used [Sun10].

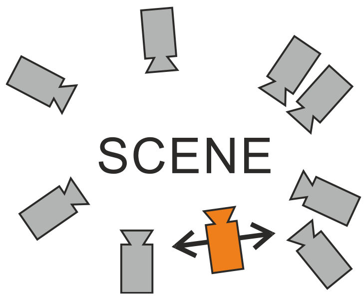

Figure 1. Idea of the immersive video system; gray – real cameras, orange – virtual camera.

There are numerous virtual view synthesis methods and algorithms described in literature (e.g. [Dzi19][Fac18][Nia18][Sen18][Wan19]). However, they cannot be used in the practical immersive video system because of the processing time. When a user of the immersive video system demands a particular virtual view, the view has to be generated immediately in order to eliminate delays between user's action and viewpoint change. Therefore, all the processing has to be performed in the real time.

The real-time virtual view synthesis methods are also known, but they usually require dedicated FPGA [Aki15][Li19][Wan12] or VLSI [Hua19] devices or powerful graphic cards [Non18][Yao16][Zha17].

In the practical, consumer immersive video system, developed for the entire spectrum of final users, it may disqualify users, as they do not have appropriate hardware due to cost or compatibility.

In this paper, the real-time virtual view synthesis for CPU is presented. So far, only one this kind of method was described in literature [Dzi18]. It was able to process FullHD sequences in real-time, but only for reduced output resolution (i.e. qHD), not to mention higher resolutions (4K). The method presented in this paper allows synthesis of UltraHD content in the real-time, what makes it usable also for the most recent immersive video systems.

2. VIEW SYNTHESIS ALGORITHM
The practical view synthesis method should meet two main requirements. At first, it has to be fast enough to be used in the real-time, consumer immersive video system. Secondly, the quality of synthesized virtual views should be as high as possible.

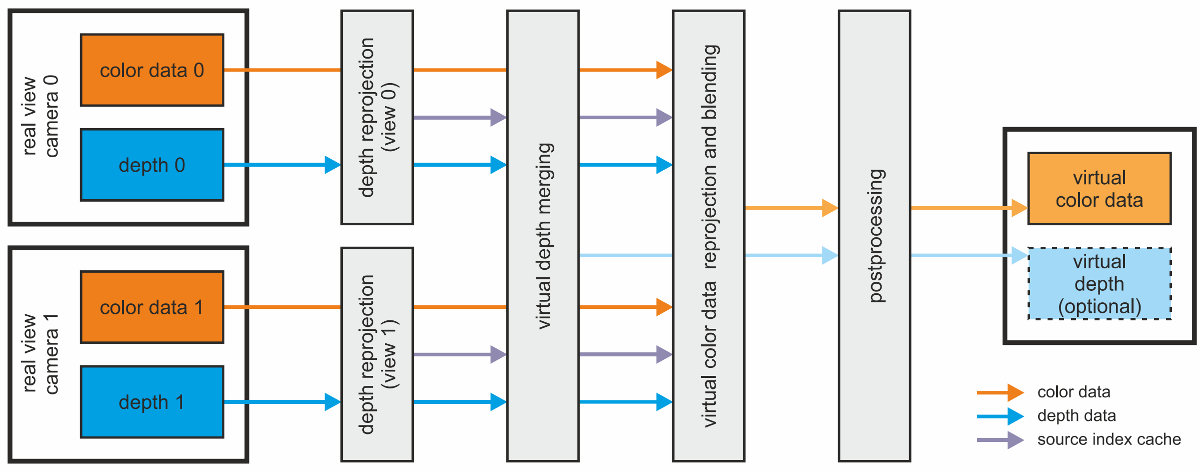

Figure 2. Block diagram presenting data flow in proposed algorithm.

In order to obtain good quality of virtual views, we decided to develop a backward-type synthesis [Duh13][Shi13]. The major advantage of such synthesis type is to admit filtering of the reprojected depth map before texture reprojection. However, typical backward-type synthesis requires two steps of reprojection: depth from input view to the virtual view and texture in the opposite direction, which makes it slower. In the proposed approach, the backward-type synthesis was modified in order to reduce the number of reprojections to one.

The proposed view synthesis algorithm consists of four main stages (Fig. 2): depth reprojection (performed separately for both real views), depth merging, color data reprojection and preprocessing. The purpose of two first operations is to create a depth map of the virtual view. Then, this depth map is used for reprojecting color data to the virtual view. Finally, the virtual view is postprocessed in order to achieve the highest quality. All the steps of proposed algorithm are described in following subsections.

Depth reprojection

In the first stage only input depth maps are analyzed. Reprojection of each pixel of the input view i is conducted by multiplication of a vector containing pixel's position $(x_i,\ y_i)$ and depth (z_i) and a homography matrix $\mathbf{H}_{i,v}$:

$$\begin{bmatrix} x_v \\ y_v \\ z_v \\ 1 \end{bmatrix} = \mathbf{H}_{i,v} \cdot \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}.$$

The homography matrix is a 4×4 matrix defined as multiplication of projection matrix of virtual view v and inverted projection matrix of input view i [Hey08].

This operation requires 16 multiplications and 12 additions for each processed pixel. In the proposed algorithm it was optimized, resulting in 4 multiplications and 3 additions per pixel, with additional 4 multiplications and 3 additions for each column and each row. However, calculations for columns and rows are performed once in a preprocessing step and their results are then stored in look up tables (LUTs).

Depth merging

In the first stage, both input views are processed separately. It results in two virtual depth maps, each containing depth values reprojected from single input depth map.

In the second stage, both virtual depth maps are merged. Within the merging operation, three cases are possible for each pixel.

If the pixel was not reprojected from any input view – it will become empty in the merged depth map. If it was reprojected from only one view – the merged depth map will contain value reprojected from that view. Finally, if there are two depth candidates for one pixel, smaller depth (closer to a camera, because it occludes further one) will be copied into the merged view.

Color data reprojection and blending

In the third stage, the actual virtual view is created. Each pixel of the virtual view is calculated by analyzing values in corresponding pixels in input views. In the typical backward-type synthesis, it would require an additional reprojection step. In the proposed approach, positions of corresponding pixels within input views are stored in source index cache, which reduces this operation to memory reading.

If the pixel was visible in only one input view, its color is copied from that input view. If it was reprojected from both input views, its color in the virtual view is calculated as an average of colors in both input views. If it was not visible in any input view, it remains a hole.

Postprocessing

The last stage of the proposed algorithm allows enhancement of the quality of the synthesized virtual view. It consists of two main steps: filtering and inpainting.

In the first step, the virtual view and corresponding depth map are filtered. They are filtered at the same time, but only depth values are being analyzed in this step. The virtual view is filtered in order to remove small artifacts such as object discontinuities or single pixels with wrong depth caused by blurred edges in input depth maps.

In order to perform fast filtration, it is checked for each pixel, whether it is surrounded by pixels with different depth. In horizontal filtering step, it is checked if the depth of analyzed pixel is much higher or much lower, than depth of its left and right. If so, color and depth of analyzed pixel is replaced by color and depth of its right. The vertical filtering step is performed based on top and bottom neighbors and filtered pixel is eventually replaced by bottom neighbor.

In the second step inpainting of the virtual view is performed. This step is crucial for the virtual view synthesis [Ber00][Cri04][Dar10] because it allows holes (i.e. areas without information reprojected from input views) filling in the virtual view. In the proposed approach the fast depth-based 4-way inpainting method is used. For each empty pixel four closest pixels in 4 directions (left, right, top and bottom neighbors) are compared. Color of the pixel with closest depth is copied to the analyzed one.

3. IMPLEMENTATION AND OPTIMIZATION DETAILS

The proposed algorithm has been implemented in portable C++ language, therefore (excluding vectorized version mentioned further) can be ported to almost any hardware platform. The single threaded implementation is based on the one described in [Dzi18]. Nonetheless, significant improvements have been developed. The proposed implementation allows processing high bit-depth sequences (up to 14 bits per pixel) and high precision depth maps (up to 16 bits per depth element).

The algorithmic optimization includes following techniques:

1. memory access optimization by reducing redundant loads/stores and using prefetch friendly data layouts,
2. usage of local buffers and pre-calculated LUTs,
3. reduction of the number of required multiplication and additional operations in depth reprojection stage.

Moreover, the implementation eliminates projection related computations from virtual view projection stage and reduce its computational complexity. During preceding (depth reprojection) stage, the source location of depth element is cached and reused in view reprojection.

Implementation using vector extensions

The majority of modern processors include some sort of vector processing units, allowing computations on several values at once. The usage of vector instructions allows a significant reduction of computation times, especially for execution bound algorithms.

In case of virtual view synthesis, depth reprojection is the most computationally complex. Therefore we decided to develop vectorized implementation of depth reprojection stage. We concentrated on AVX2 and AVX512 extensions available in modern x86-64 processors. The AVX2 and AVX512 extensions [Dem13] enable operating on 256 bit (containing 8 single precision floats) and 512 bit (containing 16 single precision floats) vector respectively. Moreover, both extensions allow to use FMA instructions [Qui07] which are very useful in the reprojection stage.

The AVX512 vector is twice as wide as AVX2 one, allowing twice as much data at one clock cycle. In addition, AVX512 instruction set admits mask registers and per-lane predication, both to write more efficient code and to reduce register pressure [Dem13].

Parallel implementation

Another approach to speed up the virtual view synthesis is to parallelize computation by using multi-threaded implementation.

Most of the synthesis-related operations, like depth merging, color data reprojection and merging and postprocessing, could be easily parallelized by dividing the picture into arbitrary number of slices and processing each slice by dedicated thread.

Unfortunately, the most complex operation in the proposed algorithm, namely the depth reprojection, is not easy to parallelize. The reason is the risk of data race caused by unpredictable location of a reprojected depth element. Therefore, there is no possibility to simply compute each of input depth slices by separate thread.

The simplest approach, presented in [Dzi18] is to perform reprojection of each depth in a separate thread i.e. the first thread processes the “depth 0”, while the second processes the “depth 1”, and so on. Unfortunately, this approach allows parallelization by factor of 2 only and is insufficient in case of modern multicore processors.

Independent Projection Targets

In order to improve the parallelization factor for depth reprojection, the Independent Projection Targets (IPT) approach has been proposed. The idea of IPT is to use separate buffers (projection targets) for each of processing threads (Figure 3). Both reprojected depth and source index cache are buffered. The usage of IPT removes the restriction for depth reprojection parallelization level and allows using all available processing threads.

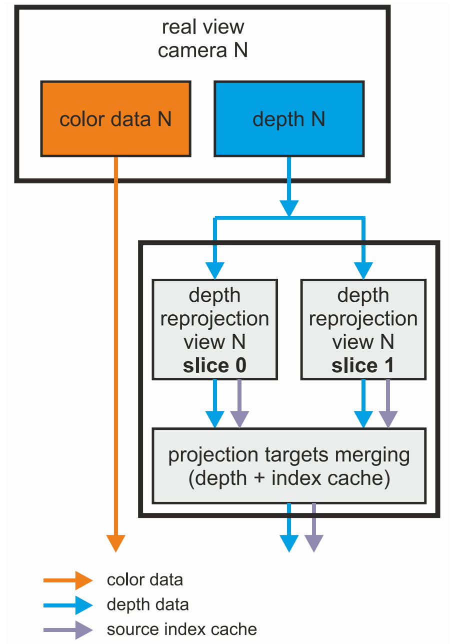

Figure 3. Independent Projection Targets.

The drawback of IPT is the necessity of additional operations to merge results from all projection targets, as well as the increase of memory footprint due to excessive buffering. Nonetheless, the additional complexity of depth merging stage does not offset the reduced complexity of depth reprojection stage.

4. METHODOLOGY

Test sequences

The test set contained three miscellaneous high-resolution test sequences (Fig. 4):

1. PoznanFencing, FullHD resolution, sparse arc arrangement [Dom16],
2. TechnicolorPainter, 2K resolution, dense linear arrangement [Doy17],
3. PoznanBasketball, FullHD resolution, sparse linear arrangement [Dom18].

Two of them (1 and 2) are commonly used in the research and developing immersive video standards. The third one was placed into the set because of very different content/characteristics – it contains a fragment of basketball match, what could be one of possible use cases of immersive video systems.

In order to simulate virtual view synthesis for UltraHD (4K) input views, one of the experiments required UltraHD sequence. Because of lack of such test material, resolution of TechnicolorPainter sequence was increased. Remaining samples of input view were calculated using 1st order interpolation, while samples of depth maps – using 0th order interpolation (in order to avoid introducing non-existent depth values at the objects' edges – if linear interpolation will be used, physical edges of the objects will be destroyed, e.g. between a pixel representing a person and a pixel representing a wall behind, there would be a pixel with averaged depth, representing physically non-existing object).

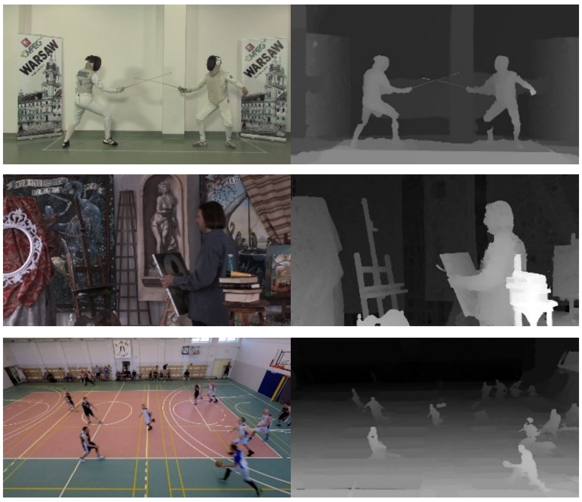

Figure 4. Input views and corresponding depth maps for (from top): PoznanFencing, TechnicolorPainter, PoznanBasketball.

Evaluated implementations

Experiments were performed on 10 implementations. Implementations were divided into 4 types: R – the reference implementation, which does not include any optimizations and is treated as a base for comparison with others; A – the optimized implementation; B – optimized and vectorized implementation using AVX2 instruction set; C – optimized and vectorized implementation using AVX512 instruction set.

Moreover, each implementation (except for reference one) was tested in 3 versions: single-threaded (1), multi-threaded (2) and multi-threaded with IPT (3).

Quality evaluation

In order to evaluate the quality of virtual views synthesized using presented algorithm, 5 objective quality metrics were used: PSNR, Multi-Scale SSIM (MS-SSIM) [Wan03], Visual Information Fidelity (VIF) [She06], Video Multimethod Assessment Fusion (VMAF) [Li16] and IVPSNR, which is ISO/IEC MPEG's metric for immersive video [MPEG19].

Implementation	Implementation features				Processing time [ms]				
	Optimized	Vectorized	Multi-threaded	Independent Projection Targets	Depth projection	Depth merging	View projection	Post-processing	Total
VSRS (state-of-the-art view synthesis method)					–	–	–	–	**2581.12**
R	–	–	–	–	127.41	0.83	14.19	18.84	**161.27**
A1	✓	–	–	–	39.25	0.82	15.81	19.30	**75.18**
A2	✓	–	✓		35.30	0.32	4.29	6.04	**45.95**
A3	✓	–	✓	✓	23.42	2.53	4.11	5.63	**35.69**
B1	✓	AVX2	–	–	15.77	0.79	10.98	18.73	**46.26**
B2	✓	AVX2	✓	–	18.62	0.32	2.62	5.47	**27.03**
B3	✓	AVX2	✓	✓	10.16	1.85	2.59	5.66	**20.26**
C1	✓	AVX512	–	–	10.26	0.80	11.04	19.03	**41.13**
C2	✓	AVX512	✓	–	12.83	0.32	2.62	5.58	**21.35**
C3	✓	AVX512	✓	✓	7.62	1.74	2.59	5.41	**17.35**

Table 1. Comparison of all implementations (TechnicolorPainter sequence, FullHD → FullHD scenario)

Test sequence			Implementation	Processing time [ms]				
Sequence name	Input / output resolution	Camera arrangement		Depth projection	Depth merging	View projection	Post-processing	Total
TechnicolorPainter	2048×1088	dense linear	C3	7.62	1.74	2.59	5.41	**17.35**
PoznanBasketball	1920×1080	sparse linear	C3	4.28	1.77	4.04	4.28	**14.37**
PoznanFencing2	1920×1080	sparse arc	C3	4.18	1.78	5.77	5.20	**16.94**

Table 2. Comparison of all test sequences (FullHD → FullHD scenario, C3 implementation)

Test sequence			Implementation	Processing time [ms]				
Sequence name	Input resolution	Output resolution		Depth projection	Depth merging	View projection	Post-processing	Total
TechnicolorPainter	4096×2176	4096×2176	C3	17.60	6.86	7.92	11.35	**43.74**
TechnicolorPainter	4096×2176	2048×1088	C3	21.68	1.71	2.72	5.08	**31.20**

Table 3. UltraHD input sequence (C3 implementation)

Quality metric	Sequence name / View synthesis algorithm					
	TechnicolorPainter		PoznanBasketball		PoznanFencing2	
	VSRS	Proposed	VSRS	Proposed	VSRS	Proposed
Y-PSNR	35.94 dB	36.69 dB	28.75 dB	29.27 dB	28.26 dB	28.88 dB
C_B-PSNR	46.81 dB	47.72 dB	40.13 dB	41.76 dB	44.72 dB	45.42 dB
C_R-PSNR	46.78 dB	47.04 dB	39.48 dB	37.08 dB	39.50 dB	44.76 dB
VIF	0.574	0.615	0.456	0.482	0.272	0.270
VMAF	87.48	91.24	59.53	61.75	56.77	57.20
MS-SSIM	0.981	0.984	0.949	0.955	0.936	0.933
IVPSNR	45.94 dB	47.56 dB	36.26 dB	36.60 dB	40.07 dB	40.29 dB

Table 4. Virtual view synthesis quality

The proposed method was compared to commonly used state-of-the-art method, developed by ISO/IEC MPEG group, namely View Synthesis Reference Software (VSRS) [Sen17].

Synthesis time evaluation

The computational complexity of each implementation was evaluated by measuring processing time. Moreover, detailed statistics for each synthesis stage have been gathered.

The calculations were performed on the desktop computer equipped with 10-core CPU based on the "Skylake-X" microarchitecture. Time measurements were made using precision time stamps according to [MDNL20].

5. EXPERIMENTAL RESULTS

Comparison of all described implementations has been presented in Table 1. The results are presented for TechnicolorPainter sequence (as the worst case of all considered sequences). In the case of reference implementation, the synthesis of virtual view frame takes ~160 ms which corresponds to ~6 frames per second (FPS). This is obviously insufficient for real-time purposes. The fastest implementation – C3 (optimized, multi-threaded and with AVX512 usage) requires only 17.35 ms to synthesize one frame (resulting in 57 FPS).

Usage of vectorized implementation allows reducing depth reprojection time from ~39 ms to ~16 ms and ~10 ms for AVX2 and AVX512 respectively.

The parallel processing significantly reduces computation time for view projection and post-processing stages. In the case of depth projection, parallel processing without IPT does not seem beneficial. Usage of IPT significantly speeds up the projection stage, however it increases the complexity of depth merging stage. Nevertheless, the IPT reduces total synthesis time.

The computation time for state-of-the-art technique (VSRS) oscillate near 2.5 seconds which makes the proposed technique two orders of magnitude faster when compared to VSRS.

Table 2 includes results for all test sequences. It is noticeable that proposed synthesizer retains its performance regardless of input sequence type. Moreover, the synthesis time for sequences with sparse camera arrangement (especially PoznanBasketball) is even shorter than for previously analyzed TechnicolorPainter.

Additional results (Table 3) have been gathered for simulated UltraHD (4K) data and measured as ~23 FPS and ~32 FPS for UltraHD and FullHD target respectively. The synthesis with UltraHD source and FullHD target resolution could be considered as typical use for transmission to mobile devices.

Comparison with state-of-the-art reference technique (VSRS) shows similar synthesized image quality for both VSRS and the proposed technique (Table 4, Fig. 5). Therefore, no quality degradation was introduced during development of fast synthesis algorithm.

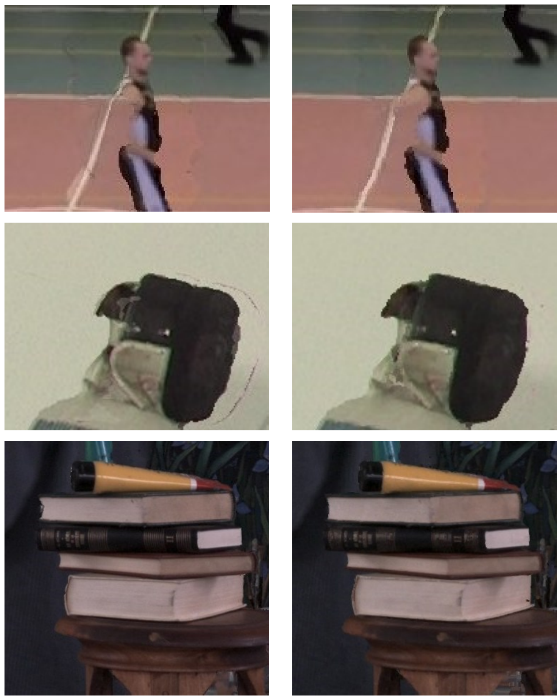

Figure 5. Fragments of virtual views synthesized using VSRS (left) and proposed method (right).

6. CONCLUSIONS

The real-time virtual view synthesis method has been presented in this paper. The experimental results show, that CPU-based implementation of the real-time view synthesis assuring good-quality virtual views is possible, even for UltraHD sequences. Therefore, it will be possible to develop cheap, consumer immersive video systems in the near future.

7. ACKNOWLEDGMENTS

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

8. 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.

[Cri04] Criminisi, A., Perez, P., and Toyama, K. Region filling and object removal by exemplar-based image inpainting. IEEE Transactions on Image Processing, vol. 13, no. 9, pp. 1200-1212, 2004.

[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.

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

[Dom16] Domański, M., Dziembowski, A., Grzelka, A., Łuczak, A., Mieloch, D., Stankiewicz, O., and Wegner, K. Multiview test video sequences for free navigation exploration obtained using pairs of cameras. ISO/IEC JTC1/SC29/WG11 MPEG, M41994, M38247, Geneva, Switzerland, 2016.

[Dom18] Domański, M., Dziembowski, A., Grajek, T., Grzelka, A., Klimaszewski, K., Mieloch, D., Ratajczak, R., Stankiewicz, O., Siast, J., Stankowski, J., and Wegner, K. Free-viewpoint Television demonstration for sports events. ISO/IEC JTC1/SC29/WG11 MPEG, M41994, Gwangju, Korea, 2018.

[Doy17] Doyen, D., Langlois, T., Vandame, B., Babon, F., Boisson, G., Sabater, N., Gendrot, R., and Schubert, A. Light field content from 16-camera rig. ISO/IEC JTC1/SC29/WG11 MPEG, M40010, Geneva, Switzerland, 2017.

[Duh13] Du-Hsiu, L., Hsueh-Ming, H., and Yu-Lun, L. Virtual view synthesis using backward depth warping algorithm. Picture Coding Symposium, PCS 2013, San Jose, 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.

[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.

[Hey08] Heyden, A., and Pollefeys, M. Multiple view geometry. In: Emerging Topics in Computer Vision. Prentice Hall, pp. 63-75, 2008.

[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.

[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.

[MPEG19] Software manual of IV-PSNR for Immersive Video. ISO/IEC JTC1/SC29/WG11 MPEG, W18709, Göteborg, Sweden, 2019.

[Nia18] Nian, Z., and Jung, C. High-quality virtual view synthesis for light field cameras using multi-loss convolutional neural networks. 2018 25th IEEE International Conference on Image Processing (ICIP), Athens, Greece, 2018.

[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.

[Qui07] Quinnell, E., Swartzlander, E.E., and Lemonds, C. Floating-point fused multiply-add architectures. 41 Conference on Signals, Systems and Computers, Pacific Grove, pp. 331-337, 2007.

[Sen17] Senoh, T., Yamamoto, K., Tetsutani, N., Yasuda, H., and Wegner, K. View Synthesis Reference Software (VSRS) 4.2 with improved inpainting and hole filing. ISO/IEC JTC1/SC29/WG11 MPEG, M40657, Hobart, Australia, 2017.

[Sen18] Senoh, T., Tetsutani, N., and Yasuda, H. Depth estimation and view synthesis for immersive media. 2018 International Conference on 3D Immersion (IC3D), Brussels, Belgium, 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.

[Shi13] Shimizu, S., Sugimoto, S., Kimata, H., and Kojima, A. Backward view synthesis prediction using virtual depth map for multiview video plus depth map coding. 2013 Visual Communications and Image Processing (VCIP), Kuching, Malaysia, 2013.

[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.

[Sun10] Sun, W., Xu, L., Au, O.C., Chui, S.H., and Kwok, C.W. An overview of free viewpoint Depth-Image-Based-Rendering (DIBR). Proceedings of the Second APSIPA Annual Summit and Conference, Biopolis, Singapore, pp. 1023-1030, 2010.

[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.

[Wan12] Wang, J., and Roeningen, L.A. Real time believable stereo and virtual view synthesis engine for autostereoscopic display. 2012 International Conference on 3D Imaging (IC3D). Liege, Belgium, 2012.

[Wan19] Wang, S., and Wang, R. Robust view synthesis in wide-baseline complex geometric environments. 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, United Kingdom, 2019.

[Yao16] Yao, L., Liu, Y., and Xu, W. Real-time virtual view synthesis using light field. EURASIP Journal on Image and Video Processing, vol. 2016, pp. 1-10, 2016.

[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.