RaViS: Real-time accelerated View Synthesizer for immersive video 6DoF VR

Daniele Bonatto, Sarah Fachada, Gauthier Lafruit
Laboratory of Image Synthesis and Analysis, Université Libre de Bruxelles; Brussels, Belgium

Abstract

MPEG-I, the upcoming standard for immersive video, has steadily explored immersive video technology for free navigation applications, where any virtual viewpoint to the scene is created using Depth Image-Based Rendering (DIBR) from any number of stationary cameras positioned around the scene. This exploration has recently evolved towards a rendering pipeline using camera feeds, as well as a standard file format, containing all information for synthesizing a virtual viewpoint to a scene. We present an acceleration of our Reference View Synthesis software (RVS) that enables the rendering in real-time of novel views in a head mounted display, hence supporting virtual reality (VR) with 6 Degrees of Freedom (6DoF) including motion parallax within a restricted viewing volume. In this paper, we explain its main engineering challenges.

Introduction

In the domain of Virtual Reality (VR) free navigation, there exists a continuum of approaches as proposed by the Milgram scale [27], exposing two extremes: on one hand, the explicit 3D polygonal model approach, typically used in 3D games, and on the other hand, the image-based approach, with 360 video as a typical representative. The former provides full 6 Degrees of Freedom (6DoF) navigation, while the latter is restricted to rotational head movements (3DoF). An intermediate approach consists of allowing 6DoF in a restricted volume, aka 3DoF+, by using Depth Image-Based Rendering (DIBR) [15].

DIBR is a topic of interest in the MPEG-I ("I"=Immersive) video activities of the worldwide MPEG standardization committee since a couple of years, seeking to find solutions for immersive video in head mounted displays, without the need of fully describing the scene geometrically with polygonal 3D primitives. In particular, for natural scenery, such explicit 3D model would be too difficult to generate, inevitably resulting in an uncanny valley effect with little cinematic quality.

Historically, the MPEG-I video expert group has first developed VSRS (View Synthesis Reference Software) [43] and associated depth estimation DERS (Depth Estimation Reference Software) [47], and in early 2019 a new Reference View Synthesizer (RVS) [12, 14] was developed (quality gains of RVS over VSRS can be found in [13]), which is a fundamental ingredient (besides of the compression add-ons) of the upcoming MPEG-I video standard, to be released in 2020 [25], aka MIV: Metadata for Immersive Video.

In this paper we present our real-time GPU implementation of RVS, aka RaViS: Real-time accelerated View Synthesizer.

Thanks to RaViS' real-time performances, we support image-based 6DoF VR experiences with DIBR without any explicit 3D modelling of the scene and objects. As such, we avoid the uncanny valley effect for natural content 6DoF VR. Moreover, we enable the creation of 3D cinematic quality content using only a small amount of RGB pictures.

Figure 1. *DIBR representation. The input images and their corresponding depth maps are used to render a new image anywhere in 3D space.*

By closing this gap, we enable users to capture scenes with standard cameras and explore their virtual copies in real-time.

Related Work

Various rendering methods exist for virtual reality. On one side, we can explore 360 degrees videos content in HMD. While reaching cinematic quality, this kind of content prevents the movement of a user in 6 Degrees-of-Freedom (6DoF) supporting only head rotations, therefore disrupting the VR experience. On the other side, we have 3D content requiring artistic knowledge and work to generate it. This content feels synthetic in VR and comes in three flavours:

1) Game engines, for example, Unity [46], and Unreal Engine 4 [11] are able to render simple to medium complex geometries in real-time. 2) High quality content similar to cinematic content can be generated using 3D modelling software such as Blender [3] by generating a scene in a cube around the user at each viewpoint in every direction on a S^2 sphere [28]. 3) Light-fields solutions use a special kind of cameras which can capture a scene from a large amount of small camera sensors, for example the Lytro camera [34], offering native novel views.

In (1), we can use techniques such as photogrammetry [36, 33] to obtain high quality 3D objects that mimic reality without the need for 3D artist to build them. They often consist in a very high number of polygons (polycount) which prevent real-time for scenes entirely based on this kind on models. Of course, (2) needs to discretize a sphere around the viewing point and the space around the end user. This kind of rendering takes from weeks to months to be generated for any frame. Furthermore, in both cases,

we need to store and transmit this data to the end user which is huge in both solutions. The main difference between (1) and (2) is the polycount and the rendering which uses ray-tracing for (2). Finally, (3) while of high quality, offers only content consisting of head rotations and small translations (3DoF+). The photorealism is close or higher to solution (2) [32].

DIBR [15, 45] is a technique that aims to generate novel views of a scene using a number of fixed input images and their corresponding depth maps. Given a novel viewpoint to generate, DIBR displaces the colors from the input images to their new location in accordance to the depth maps. For example, in Figure 1 we illustrate two input views with their associated depth maps and in the center a projection of the color content in 3D. While the two images are projected separately into the final position, their blending results in a high quality view synthesis with little disocclusions.

DIBR with one input image has been attempted in CPU taking several seconds to minutes to render one frame [43, 30] and some GPU implementations exists [30, 8, 37]. However, employing only one input view as entry limits the navigability of a scene as it lacks depth information around the main viewpoint, resulting in a lot of disocclusions near the borders of the objects.

On the other side of the DIBR spectrum lies Light Field (LF) rendering techniques where the plenoptic function of a scene is sampled using a large number of imaging sensors called lenslet arrays [48]. However, the plenoptic function is a dense function in space, and thus acquiring a picture requires several lenses very close to each other. By re-sampling the obtained light field from a camera, we can reconstruct the scene from any viewpoint in the sensor total baseline [48]. LF obtains very high quality outputs for small scene displacements. A recent attempt from Google [32] produced very high quality outputs in head mounted displays, but their data processing workflow is extremely slow and needs expensive hardware to produce a very limited head movement.

All DIBR techniques are dependent on the depth quality. Several techniques of depth estimation are used depending on the type of input views. In practice we use a depth estimation software for multi-view camera setups and LF, which solves the correspondence problem and constructs a depth map from the input views.

Methodology

For the sake of clarity, we develop how one input view is warped to its final place and then we will generalize on n images and how we achieve this exploiting the GPU parallelization. Our procedure is described in Figure 1. First, we describe what the needed inputs for our algorithm are, how to project the textures to their final position and finally how to blend them.

An input view in DIBR refers to a texture (the image captured by a camera) and a depth map with its calibration data. Usually, for natural scenes, we only have access to the textures datasets without proper calibration data (Section Calibration).

To synthesize novel views, DIBR uses the depth information and the calibration parameters to displace the colored pixels in 3D to a virtual position. Given an input and the virtual pose, we warp the colors of the texture to their final position by mathematically projecting and unprojecting them using the extrinsic matrices and the depth map.

Mapping each color pixel to its final destination results in a sparse colored point cloud. To avoid meshing the point cloud - for example with Poisson reconstruction [20], which is extremely compute intensive- the input texture is directly meshed by building triangles between adjacent pixels, interpreted as vertices in space thanks to their depth information. By continuous deformation, those triangles are warped during the projection and unprojection step. We then leverage the OpenGL rasterization to fill them by interpolating between the colors. This method avoids occlusions and is essentially cost free. However, interfaces between objects give elongated triangles due to the high depth variation. In the blending procedure, we discard those elongated triangles (Figure 6(a)), resulting in more pleasing images as in Figure 6(b). In Fig. 2(b) bottom center image, we observe the discarded triangles as a hole in the chair. This last step avoids to mesh the 3D point cloud, and inpaints the smallest disocclusions. Furthermore, this step is done in the vertex shader, avoiding unnecessary, per fragment, computations.

When all the views are synthesized in their final position, we blend them together by averaging them. Depending on which input views are used, several unavoidable disocclusions can appear as in Fig. 2(b) bottom right, where the information was simply missing over all the input views. But, in average, a lot of disocclusions are filled. We observe some blending defects in small regions such as in Fig. 2(b) bottom left.

The blending takes part in two specialized Framebuffer Objects (FBO) that are in charge of accumulating the colors. The first FBO stores the current generated view, while the second FBO stores the blended views. We use these two FBO as ping-pong buffers to limit the memory usage.

To render in 3D, the HMD pose is used to generate two virtual camera views approximately 6 cm apart for the two eyes of the user.

The number of input views matters to avoid disocclusions, however, for planar datasets, where every image is in the same plane, adding novel views does not improve these disocclusions. We noted that taking 4 input views around the HMD position gives good results without slowing down the process. This is obviously dependent on the dataset. In the first row of Figure 2, three different viewpoints of a scene generated from only 4 cameras can be observed. In the second row, several disocclusions can be observed.

Depth Maps

The quality of the depth maps directly impacts the final synthesized image. 3D content offers automatically depth maps associated with any frame. This luxury does not exist in natural scenes. Depending on the texture acquisition, several techniques are used to obtain good quality depth maps. We can distinguish, RGB cameras, Active Depth acquisition sensors and Light-Field Imagery.

For RGB captured images, we do not have any information about the depth; for planar and linear camera arrangements several Stereo-matching techniques exists and are commonly used. While they are often slow, they offer high quality outputs. They consist in solving the corresponding problem between every two input views and proceeding with a global non-parallelizable optimization step [24, 40].

For cameras in a general position, we can find features in the images and match them to estimate the cameras relative po-

Figure 2. *Classroom dataset. Top row: Synthesized views using 4 input views and rotating around a central point in the scene. Bottom row: Zoomed in details of a bad blending (hardly visible at full scale), a small disocclusion and a full disocclusion. All three problems are due to missing information (i.e. information not captured by any camera).*

sition and rotation. Then the images are projected to a common projection plane to run a stereo matching algorithm between the rectified views.

Active sensing cameras (such as Microsoft Kinect [49]) use structured light, random light pattern matching, or Time-of-Flight technologies to obtain the depth map associated with the image. Those RGB-D cameras generate accurate depth maps for the image. However, the depth sensor is usually not aligned with the RGB sensor and the output resolution is often far less than the RGB resolution. Depth scaling and inpainting need to be performed to obtain high quality images as in [41].

For Light-Field imagery (Lytro), we obtain a sample of the plenoptic function at a given time. By resampling it, we can generate multiple input views with different small baselines for static scenery. When we obtain those sampled views, they are in a 2D plane, and the problem reduces to stereo matching in any direction. Several new methods use multiple light field cameras to do multi-scale stereo matching [39, 38].

In the present paper, we mainly used DERS [47], the Depth Estimation Reference Software from the MPEG community, which performs depth estimation with 4 input views in any spatial pose. DERS has advanced features such as a Graph-Cut optimization to refine the depth maps and temporal enhancement to speed up the computations by using multiple frames in dynamic content. We also used RDE (Reference Depth Estimation) software [40], the novel version of DERS which extends it by taking any number of input views.

Results and Discussion

Experiments were done on a PC with Intel Xeon E5-2680@2.7GHz CPU and NVIDIA GTX 1080TI GPU, following natural head movements on a Oculus Rift HMD, as shown in Figure 4 for the heads movements of Figure 3. We achieve 90 FPS real-time performance for 6DoF stereoscopic VR with DIBR based on 4 full-HD reference views. Each frame consists of two rendering passes, one for each eye; the maximum time per frame being 5 ms.

Several datasets were used to have a representative landscape. 3D content with perfect depth maps (eg: Classroom), real scenery with Kinect depth map (eg: ULB Unicorn) and stereo-matching estimated depth maps (eg: ULB BabyUnicorn) were used. A review of the different datasets is out of the scope of this paper and can be found in [41].

Moreover, care has been taken to obtain robustness against depth errors by using the Poznan Depth Refinement tool (PDR) [10, 6]. Resulting depth maps are shown in Figure 3. For instance, for a scene at 1.5 m distance, we use a color camera array with optical axes separation of 20 x 30 cm. 2x2 depth maps are estimated, cf. Figure 3(a); they are clearly imperfect. Nevertheless, with RaViS we obtain the high-quality synthesized virtual view of Figure 3(b), corresponding to the central position in the 2x2 camera plane; a video can be found on [5]. A video demo, corresponding to Figure 3(c) using RaViS, can be found in [4].

The position of the head can be retrieved in real-time as shown in the pose trace of Figure 4 where a selected trajectory was designed in order to analyze the output views during various natural user movements.

Technicalities

Software

The software is coded in C++11 using OpenCV [31] to handle the textures and depth map images. Custom code is made in

Figure 3. *(a) 4×4 depth maps, (b) synthesized virtual view with DIBR/RaViS, (c) real-time, image-based 6DoF VR*

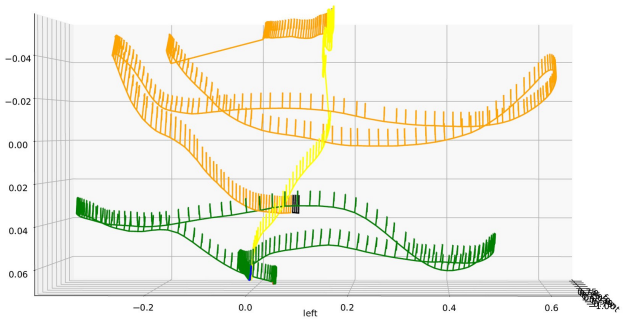

Figure 4. *Pose traces when wearing an Oculus Rift HMD corresponding to Figure 3. The three colors correspond to the three phases of the navigation: left-right movements, forward movements and left-right movements closer to the scene.*

Figure 5. *Unicorn dataset. All the synthesized views use 4 inputs. From left to right: VSRS, real image, our method described in this paper. The loss of quality comes from the depth map quality. In Figure 7(b) we can observe our results with better depth maps extracted from 8 input views.*

order to load any YUV format file and a custom Json parser is used to parse the configuration files. An example of configuration files can be found in Section Calibration. While there are many commercially available HMDs, we wanted to focus on HTC Vive and Oculus Rift as they are two of the most popular headsets. To interact with the HMD, we needed to choose between OSVR [35] an open-source solution for virtual reality and OpenVR [44] a closed-source library from Valve. At the time, OSVR was not advanced enough and several major bugs existed. OpenVR was therefore chosen.Our view synthesis method is refactored in the new MPEG-I standard, MIV.

File compression

Historically, MPEG uses YUV file format with 4:2:0 chroma subsampling for static and dynamic content. A YUV file stores images in one lumina channel (Y) and two subsampled chromina channels (UV). Depending on the subformat, the chroma channels can be scaled down as they convey less visual information. This format is easy to parse but has no compression at all. As the limiting factor is the size of the VRAM, the number of loaded textures and depth in RaViS is limited by their total size. One 1920×1080 frame in a YUV420p8le format takes $1920 \times 1080 \times (1 + 1/2 \times 2) \times 8bits \approx 4MB$ without depending on the content of the image.

In RaViS, static content is loaded in YUV or PNG file format indifferently while dynamic content (texture and depth) is encoded with a H.256 codec in a pre-processing step, yielding a high compression and input views with small video bitstreams. Each time the movie is read, the different input views are decoded in real time exploiting the CUDA decoders of the GPU [29].

In the standardization process, MPEG will exploit the redundant information of the textures and depth maps in the different input views to generate texture atlases in YUV format, which will be smaller than all the views coded independently. Furthermore, for more than one frame content, they can exploit the information from previous frames to compress the information similarly to the H.256 codec. Further details can be found in [25].

Conclusion

We presented here a real-time view synthesis software delivering 90 FPS using three to four input views in a head mounted display. The strongest points are its ability to synthesize virtual views from a minimal set of lightweight (small filesize/bitstream) input views. The depth estimation needs to be improved, but several approaches give already good results depending on the acquisition technology used. Its adoption by the MPEG-I community shows a sustained interest on this kind of technologies. Several improvements still require our attention, such as, further com-

(a) (b)

Figure 6. *Museum dataset, 4 blended input views. (a) Without discarding the elongated triangles. (b) Discarding the triangles results in few disocclusions depending on the input cameras' pose. Deep black regions are disocclusions that are not covered by any input camera view and are left unpainted.*

(a)

Figure 7. *View synthesis in HMD resolution from 8 input views. Dataset: ULB Unicorn.*

pression to transmit the data over internet and better handling of occlusions.

Annexes

Calibration

Calibration data is divided in intrinsic and extrinsic matrices. While intrinsic values depend on the camera (focal, central point), extrinsic ones depend on the pose of the camera in the scene (rotation, translation). The extrinsic matrix contains 12 values that need to be estimated. The pose estimation problem is well known [19, 26]. For small datasets, classical OpenCV calibration algorithms are used, while with large datasets, photogrammetry [36] can be used to obtain good estimated parameters by refining the values using all the images in the dataset. In section Depth Maps we discuss how depth maps can be computed using the textures.

File Formats

A unified file format is designed in order to select which are the input files, where to find the files and which view needs to be synthesized. The format is described in [23]. An example is shown in Figure 8 and its associated cameras file in Figure 9.

Acknowledgements

This work uses copyrighted materials: *Unicorn dataset* [7] created in the 3DLicorneA project, supported by Innoviris the Brussels Institute for Research and Innovation Belgium, under contract No.: 2015-DS-39a/b/c/d, 3DLicorneA. ULB BabyUnicorn dataset [41]. *Technicolor Museum*, Technicolor. All rights reserved Copyright© 2017-2018 [9] and *Classroom dataset* [22].

```
{
  "Version": "2.0",
  "InputCameraParameterFile":
    "./cameras.json",
  "VirtualCameraParameterFile":
    "./cameras.json",
  "InputCameraNames": [
    "cam_name_01",
    "cam_name_02",
    ...
  ],
  "VirtualCameraNames": ["cam_name_XY"],
  "ViewImageNames": [
    "./dataset/texture_000001.yuv",
    "./dataset/texture_000002.yuv",
    ...
  ],
  "DepthMapNames": [
    "./dataset/depth_000001.yuv",
    "./dataset/depth_000002.yuv",
    ...
  ],
  "OutputFiles": [
    "Texture_out_cam_0000XY.png"
  ],
  "StartFrame": 0,
  "NumberOfFrames": 1,
  "Precision": 2.0,
  "ColorSpace": "RGB",
  "ViewSynthesisMethod": "Triangles",
  "BlendingMethod": "Multispectral",
  "BlendingLowFreqFactor": 1.0,
  "BlendingHighFreqFactor": 4.0
}
```

Figure 8. *experiment.json, a configuration for an experiment.*

```
{
  "Version": "1.0",
  "cameras": [{
      "Name": "cam_name",
      "Position": [px, py, pz],
      "Rotation": [Rx, Ry, Rz],
      "Depth_range": [650, 1550],
      "Resolution": [1920, 1080],
      "Projection": "Perspective",
      "Focal": [fx, fy],
      "Principle_point":
        [pp_x, pp_y],
      "BitDepthColor": 8,
      "BitDepthDepth": 16,
      "ColorSpace": "YUV420",
      "DepthColorSpace": "YUV400"
    },
    {...}]
}
```

Figure 9. *cameras.json, pose and format of all cameras in a dataset*

References

[1] Valerie Allie, Bart Kroon, and Lu Yu. Workshop on coding technologies for immersive audio/visual experiences, July 2019. https://mpeg.chiariglione.org/about/events/workshop-coding-technologies-immersive-audiovisual-experiences.

[2] Angelo V. Arecchi, Tahar Messadi, and R. John Koshel. *Field Guide to Illumination*. SPIE, August 2007.

[3] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam, 2019. http://www.blender.org.

[4] Daniele Bonatto, Sarah Fachada, and Gauthier Lafruit. RaViS: Real-time Accelerated VIew Synthesis: Depth Image-Based Rendering (DIBR), 2019. https://dipot.ulb.ac.be/dspace/bitstream/2013/291526/5/demo_unicorn_4views_1920x1080.mp4.

[5] Daniele Bonatto, Sarah Fachada, Arnaud Schenkel, and Gauthier Lafruit. RVS View Synthesis with view-coherent depth maps and DIBR, 2019. https://dipot.ulb.ac.be/dspace/bitstream/2013/294240/3/DIBR_ULB_LISA.zip.

[6] Daniele Bonatto, Arnaud Schenkel, and Gauthier Lafruit. [MPEG-I Visual] View-consistent depth maps for ULB_babyunicorn sequence [m50797]. Switzerland, Geneva, October 2019.

[7] Daniele Bonatto, Arnaud Schenkel, Tim Lenertz, Yan Li, and Gauthier Lafruit. ULB High Density 2d/3d Camera Array data set, version 2 [M41083]. *ISO/IEC JTC1/SC29/WG11*, July 2017.

[8] Roberto Gerson de Albuquerque Azevedo, Fernando Ismério, Alberto Barbosa Raposo, and Luiz Fernando Gomes Soares. Real-Time Depth-Image-Based Rendering for 3dtv Using OpenCL. In George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Ryan McMahan, Jason Jerald, Hui Zhang, Steven M. Drucker, Chandra Kambhamettu, Maha El Choubassi, Zhigang Deng, and Mark Carlson, editors, *Advances in Visual Computing*, volume 8887, pages 97–106. Springer International Publishing, Cham, 2014.

[9] Renaud Doré, Gerard Briand, and Thierry Tapie. Technicolor 3dof-plus Test Materials [M42349]. *ISO/IEC JTC1/SC29/WG11*, page 8, April 2018.

[10] Adrian Dziembowski. Manual of depth refinement software PDR [w18708]. Sweden, Gothenburg, July 2019.

[11] Epic Games. Unreal Engine 4, September 2019. https://www.unrealengine.com/.

[12] Sarah Fachada, Daniele Bonatto, Arnaud Schenkel, and Gauthier Lafruit. Depth Image-Based View Synthesis with Multiple Reference Views for Virtual Reality. In *2018 - 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, pages 1–4, Helsinki, June 2018. IEEE.

[13] Sarah Fachada, Daniele Bonatto, Arnaud Schenkel, and Gauthier Lafruit. Free Navigation in Natural Scenery With DIBR: RVS and VSRS in MPEG-I Standardization. In *2018 International Conference on 3D Immersion (IC3D)*, pages 1–6, Brussels, Belgium, December 2018. IEEE.

[14] Sarah Fachada, Bart Kroon, Daniele Bonatto, Bart Sonneveldt, and Gauthier Lafruit. Reference View Synthesizer (RVS) 2.0 manual, [N17759]. Technical report, ISO/IEC JTC1/SC29/WG11, July 2018.

[15] Christoph Fehn. Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3d-TV. In *Stereoscopic Displays and Virtual Reality Systems XI*, volume 5291, pages 93–105. International Society for Optics and Photonics, May 2004.

[16] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. DeepView: View Synthesis With Learned Gradient Descent. page 10,

July 2019.

[17] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. DeepStereo: Learning to Predict New Views from the World's Imagery. *arXiv:1506.06825 [cs]*, June 2015. arXiv: 1506.06825.

[18] Shir Gur and Lior Wolf. Single Image Depth Estimation Trained via Depth From Defocus Cues. page 10, 2019.

[19] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. 2004. OCLC: 171123855.

[20] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson Surface Reconstruction, 2006.

[21] Vladimir Kolmogorov and Ramin Zabih. Multi-camera Scene Reconstruction via Graph Cuts. In *Proceedings of the 7th European Conference on Computer Vision-Part III*, ECCV '02, pages 82–96, Berlin, Heidelberg, 2002. Springer-Verlag.

[22] Bart Kroon. 3dof+ test sequence ClassroomVideo [M42415]. *ISO/IEC JTC1/SC29/WG11*, April 2018.

[23] Bart Kroon. Reference View Synthesizer (RVS) manual [N18068]. *ISO/IEC JTC1/SC29/WG11*, page 19, October 2018.

[24] Deepika Kumari, Kamaljit Kaur, and Kamaljit Kaur. A Survey on Stereo Matching Techniques for 3d Vision in Image Processing. *International Journal of Engineering and Manufacturing*, 6(4):40–49, July 2016.

[25] Gauthier Lafruit, Daniele Bonatto, Christian Tulvan, Marius Preda, and Lu Yu. Understanding MPEG-I Coding Standardization in Immersive VR/AR Applications. *SMPTE Motion Imaging Journal*, 128(10):33–39, November 2019.

[26] Yi Ma. *An invitation to 3-D vision: from images to geometric models*. Number 26 in Interdisciplinary Applied Mathematics Imaging, vision, and graphics. Springer, New York, NY, nachdr. edition, 2010. OCLC: 846177902.

[27] Paul Milgram, Haruo Takemura, Akira Utsumi, and Fumio Kishino. Augmented reality: a class of displays on the reality-virtuality continuum. pages 282–292, Boston, MA, December 1995.

[28] Nozon. Nozon, November 2019. https://www.nozon.com/.

[29] NVIDIA. NVIDIA Video Codec SDK, August 2013. https://developer.nvidia.com/nvidia-video-codec-sdk.

[30] J Ogniewski. High-Quality Real-Time Depth-Image-Based-Rendering. page 8, 2017.

[31] OpenCV. OpenCV - Open Source Computer Vision Library, November 2019. https://github.com/opencv/opencv.

[32] Ryan S. Overbeck, Daniel Erickson, Daniel Evangelakos, and Paul Debevec. Welcome to Light Fields. In *ACM SIGGRAPH 2018 Virtual, Augmented, and Mixed Reality*, SIGGRAPH '18, pages 32:1–32:1, New York, NY, USA, 2018. ACM. event-place: Vancouver, British Columbia, Canada.

[33] Quixel. Quixel Megascans Library, 2019. https://quixel.com/.

[34] Raytrix. Raytrix, 2019. https://raytrix.de/.

[35] Razer. OSVR - Open-Source Virtual Reality, 2019. http://www.osvr.org/.

[36] RealityCapture. RealityCapture, November 2019. https://www.capturingreality.com/.

[37] Bernhard Reinert, Johannes Kopf, Tobias Ritschel, Eduardo Cuervo, David Chu, and Hans-Peter Seidel. Proxy-guided Image-based Rendering for Mobile Devices. *Computer Graphics Forum*, 35(7):353–362, October 2016.

[38] Segolene Rogge, Beerend Ceulemans, Quentin Bolsee, and Adrian Munteanu. Multi-stereo Matching for Light Field Camera Arrays. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 251–255, Rome, September 2018. IEEE.

[39] Segolene Rogge and Adrian Munteanu. Depth Estimation In Light Field Camera Arrays Based On Multi-Stereo Matching and Belief Propagation. In *2018 - 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, pages 1–4, Helsinki, June 2018. IEEE.

[40] Ségolène Rogge, Daniele Bonatto, Jaime Sancho, Rubén Salvador, Eduardo Juarez, Adrian Munteanu, and Gauthier Lafruit. MPEG-I Depth Estimation Reference Software. *2019 International Conference on 3D Immersion (IC3D)*, 2019. http://pcs2019.org/.

[41] Arnaud Schenkel, Daniele Bonatto, Sarah Fachada, Henry-Louis Guillaume, and Gauthier Lafruit. Natural Scenes Datasets for Exploration In 6dof Navigation. In *2018 International Conference on 3D Immersion (IC3D)*, pages 1–8, Brussels, Belgium, December 2018. IEEE.

[42] Bart Sonneveldt and Bart Kroon. Modify synthesizer parameters for TMIV (Test Model for Immersive Video) [M51371]. *ISO/IEC JTC1/SC29/WG11*, October 2019.

[43] Olgierd Stankiewicz, Krzysztof Wegner, Masayuki Tanimoto, and Marek Domański. Enhanced View Synthesis Reference Software (VSRS) for Free-viewpoint television [M31520]. *ISO/IEC JTC1/SC29/WG11*, January 2013.

[44] SteamVR. OpenVR, November 2019. https://github.com/ValveSoftware/openvr.

[45] Wenxiu Sun, Lingfeng Xu, Oscar C Au, Sung Him Chui, and Chun Wing Kwok. An overview of free viewpoint Depth-Image-Based Rendering (DIBR). page 8, December 2010.

[46] Unity Technologies. Unity, April 2019. https://unity.com/.

[47] Krzysztof Wegner. Software Manual DERS. page 17, 2014.

[48] Cha Zhang and Tsuhan Chen. *Light field sampling*. Morgan & Claypool Publishers, San Rafael, Calif., 2006. OCLC: 80767643.

[49] Zhengyou Zhang. Microsoft Kinect Sensor and Its Effect. *IEEE MultiMedia*, 19(2):4–10, April 2012.

Authors Biography

Daniele Bonatto received a computational intelligence software engineering degree in applied sciences from the Université Libre de Bruxelles (ULB, 2016). He is pursuing a PhD program jointly between the ULB and the Vrije Universiteit Brussel (VUB). He works on realtime free-viewpoint rendering of natural scenery with sparse multicamera acquisition setups. Jointly with the Moving Picture Experts Group (MPEG), Bonatto developed the reference view synthesis software (2018) and two high-density static and dynamic natural scene datasets.

Sarah Fachada graduated from Ecole polytechnique (France) and Trinity College of Dublin (Ireland) in 2017, majoring in computer science. She is now a PhD student at Université Libre de Bruxelles (Belgium), working on acquisition and rendering in light fields and DIBR. Her work explores fields such as rendering with non-pinhole cameras, geometric algebra applications and rendering non-lambertian objects. Jointly with MPEG, Fachada developed the reference view synthesis software (2018) and dynamic natural scene datasets.

Gauthier Lafruit received his Master of Engineering degree in Electromechanics at the Free University of Brussels, Belgium, in 1989, and his PhD in 1995 in the field of wavelet imaging. In 1996, he joined IMEC, specializing in compression and image analysis for space applications and broadcasting. This gradually led him to follow standardization committees of ESA (CCSDS), as well as audiovisual and multimedia standards in general (JPEG,

MPEG). After a year at the University of Hasselt, Belgium, he joined in 2014 the Université Libre de Bruxelles, Belgium, where he is currently professor in 3D imaging in all its forms: stereoscopy, multi-camera acquisitions, 3D video games, virtual reality and digital holography.