IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received October 7, 2021, accepted October 19, 2021, date of publication October 27, 2021, date of current version November 4, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3123529

Real-Time Depth Video-Based Rendering for 6-DoF HMD Navigation and Light Field Displays

DANIELE BONATTO[1,2], SARAH FACHADA[1], (Student Member, IEEE), SÉGOLÈNE ROGGE[2], ADRIAN MUNTEANU[2], (Member, IEEE), AND GAUTHIER LAFRUIT[1], (Member, IEEE)
[1]Laboratory of Image Synthesis and Analysis (LISA), Université Libre de Bruxelles, 1050 Brussels, Belgium
[2]Department of Electronics and Informatics (ETRO), Vrije Universiteit Brussels, 1050 Brussels, Belgium

Corresponding author: Gauthier Lafruit (gauthier.lafruit@ulb.be)

This work was supported in part by the Fonds de la Recherche Scientifque - FNRS, Belgium, project ColibriH, under Grant 33679514; and in part by the project HoviTron that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N 951989.

ABSTRACT This paper presents a novel approach to provide immersive free navigation with 6 Degrees of Freedom in real-time for natural and virtual scenery, for both static and dynamic content. Stemming from the state-of-the-art in Depth Image-Based Rendering and the OpenGL pipeline, this new View Synthesis method achieves free navigation at up to 90 FPS and can take any number of input views with their corresponding depth maps as priors. Video content can be played thanks to GPU decompression, supporting free navigation with full parallax in real-time. To render a novel viewpoint, each selected input view is warped using the camera pose and associated depth map, using an implicit 3D representation. The warped views are then blended all together to generate the chosen virtual view. Various view blending approaches specifically designed to avoid visual artifacts are compared. Using as few as four input views appears to be an optimal trade-off between computation time and quality, allowing to synthesize high-quality stereoscopic views in real-time, offering a genuine immersive virtual reality experience. Additionally, the proposed approach provides high-quality rendering of a 3D scenery on holographic light field displays. Our results are comparable - objectively and subjectively - to the state of the art view synthesis tools NeRF and LLFF, while maintaining an overall lower complexity and real-time rendering.

INDEX TERMS Virtual reality, stereo image processing, stereo vision, free viewpoint navigation, reference view synthesizer, real-time view synthesis.

I. INTRODUCTION

Rendering a scene from any viewpoint has a long history, starting from the seminal implementations of QuickTime VR [5] a quarter of a century ago, followed by the Free-Viewpoint TV activity [6] that culminated into what is called today MPEG Immersive Video (MIV), soon to be promoted to an MPEG-I standard (''I'' stands for ''immersive'') by end-2021. However, 6 degrees of freedom (6-DoF) navigation in natural content has stagnated due to inherent difficulties, going from the capture of multiview content to depth estimation and view synthesis. A robust framework enabling high-quality seamless navigation in natural scenery is yet to be created.

View generation for natural scenery has recently been revived with the advent of virtual reality (VR) applications and light field displays [7]. Many media industries are assiduously working to expand the usability of such technologies to a wider public. In this context, novel view synthesis methods are needed to allow rendering of multiview content to the consumer directly, seamlessly navigating through such content in support of 6-DoF VR, to substantially reduce bandwidth in streaming applications, and to enable high-quality 3D rendering on light field displays. Such view synthesis methods are the missing technological component to enable a large-scale deployment of 6-DoF virtual reality and glasses-free 3D displays at affordable price.

Various approaches can be followed to render natural scenery from any viewpoint in a static or dynamic context, using a dataset acquired with fixed camera poses.

For static scenes, explicit 3D models can be obtained thanks to structure from motion, by triangulation of matched features across tens, hundreds or thousands of input images [8]–[11]. However, rendering novel viewpoints from

The associate editor coordinating the review of this manuscript and approving it for publication was Jiachen Yang.

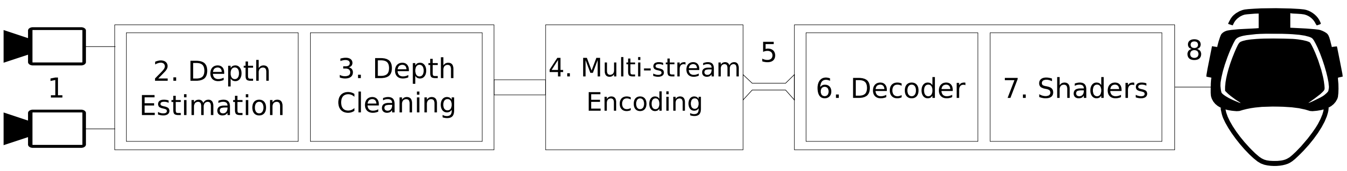


FIGURE 1. **Capturing to rendering pipeline. 1. MPEG-I input sequences. 2-3. Depth generation (Section III-B), 4. Multi-stream encoding. 5. Restricted bandwidth (Oculus Quest 2), depth images take less bandwidth/bitrate than RGB images. 6-7. Decoding (Section III-B3). 8. Quality evaluation for virtual reality. Steps 1-4 can be done offline in a pre-processing step, but shall be sufficiently rapid (minutes). Steps 5-8 can be used for an embedded HMD.**

a reconstructed scene is a difficult photo-realistic task, due to the mesh irregularities and low details. Point-cloud (PC) splatting [12], [13] can improve the quality of the obtained rendering but still lacks the photo-realism of image-based rendering (IBR) methods.

IBR techniques use the color values of the acquired images to recover the light rays appearance. The light field parametrization [14] stores the light rays localisation and orientation instead of the scene structure, which remains implicit for rendering. Similarly, most IBR methods use an implicit representation of the scene geometry: depth maps for depth image-based rendering (DIBR) [15] or an explicit one as in the geometric proxy used in the Unstructured Lumigraph Rendering (ULR) approach [16]–[18] and Stable View Synthesis [19].

More recently, Google has brought IBR back to the stage with a slow (several minutes per frame) but highly photo-realistic deep learning stereo algorithm [20]. Another light field approach, also by Google, runs in real-time and is suited for VR but at the cost of expensive hardware, heavy preprocessing and custom compression [21]. Both methods offer a 6-DoF experience limited around the acquisition point. High-quality rendering results can still be obtained with sparse inputs using preprocessing steps [22]. The limitations that this method encounters are similar to those relying on light fields [14], that is, ghosting artifacts and limited motion.

Closer to DIBR, Multiplane Images (MPI) reach fast high-quality results [23]–[26]. Each input image is separated into sparse images associated to a depth value, and those sparse images are warped to novel viewpoints before being merged.

In this paper, we also compare our DIBR results to Local Light Field Fusion (LLFF) [25], which uses deep learning to estimate the implicit geometry and the blending of the reprojected images.

Deep learning is indeed omnipresent in view synthesis, adapted to specific problems, for example interior images [27], [28], using a geometry based approach to synthesize indoor scenes by generating a global mesh and refining it with local depth map estimations. An original use of deep learning is made in NeRF [29], as the network is trained per dataset to estimate a volumetric representation of the scene. Though closer to point-clouds representations than IBR, volumetric rendering performs incredibly well in rendering natural scenes [29], [30]. In this scope, we also compare our results to NeRF [29]. Stable View Synthesis [19] uses deep learning to encode the input

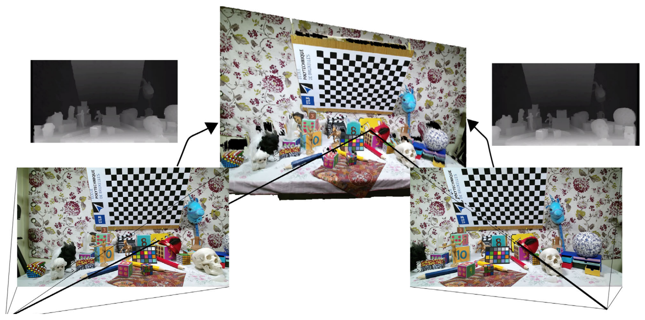


FIGURE 2. **DIBR representation. The input images and their corresponding depth maps are used to render a new image anywhere in 3D space.**

images as feature vectors on a surface of a geometric proxy.

Due to the various trade-offs between those techniques and the necessity to render video content in real-time, we have chosen to build on Depth Image-Based Rendering (DIBR) [15], [31]–[34] which -as shown in the present paper-offers a fast and accurate approach to render synthetic and natural scenes at high-quality, in real-time, directly on a head mounted display (HMD) by means of efficient usage of the available computational resources. At the same time it is also suitable for low bandwidth applications such as embedded HMD, where the data to be transmitted consists of the RGB images and their associated depth maps (efficiently encoded using image atlases [35], [36]) while the processing is done within the embedded GPU (steps 5-8 in Figure 1).

In this paper we also present an approach to accelerate our DIBR algorithm on the GPU taking several input views in any configuration, to render a scene in a HMD and/or a light field display. In contrast to many DIBR algorithms [20], [31], [32], taking only two views as input (left and right as in Figure 2), the number of input views in our approach is unlimited (and we show high-quality results with only 4 input views). This allows to address more disocclusions, following the example of light fields which sample the plenoptic function in many positions. At the same time, our approach permits a wider baseline between the input views than in the light field approach, hence covering a larger field of view in 6-DoF. To achieve this goal, we started from the MPEG-I Reference View Synthesizer (RVS) source code [37] we co-developed and distributed under BSD license,[1] and implemented proprietary shaders [38] in OpenGL to synthesize views in real-time.

[1]https://gitlab.com/mpeg-i-visual/rvs

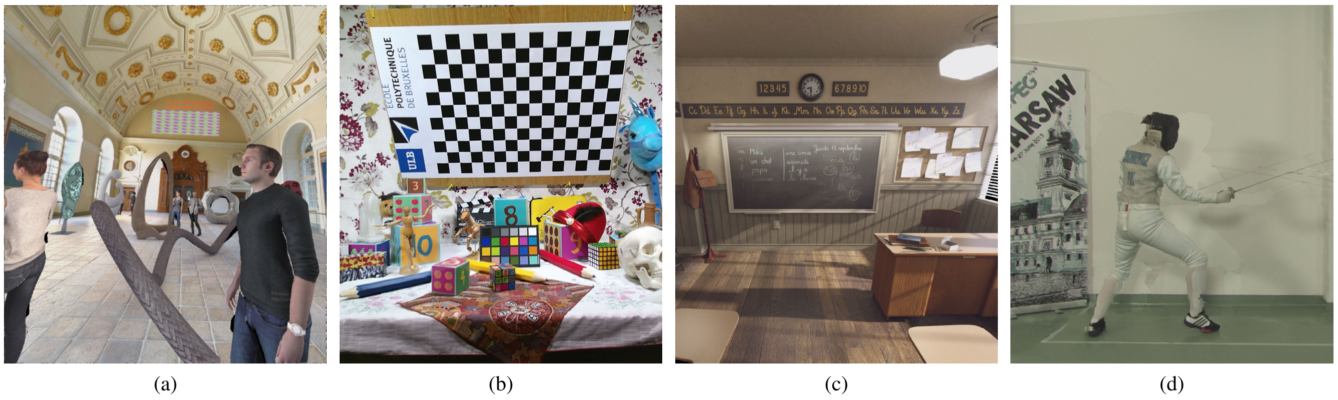


FIGURE 3. View synthesis in HMD resolution, using 8 input views for (a)-(c) and only 2 for (d), tested on four datasets: (a) Museum, (b) Toystable, (c) Classroom, (d) Fencing.

Even though our approach does not restrict the number of input views, it generates very pleasing visual results even with a very limited number of views (one to four views). This is essential for enabling broad deployment of 6-DoF VR on common GPU hardware. Some view synthesis results of the proposed method are shown in Figure 3.

Our main contributions presented in this paper are:

- A novel methodology for virtual view synthesis using a small number of input views and achieving real-time rendering on common hardware.
- Achieving high-fidelity and robust free-viewpoint rendering for 6-DoF VR applications, including large step-in (forward) and step-out (backward) movements and full motion parallax.
- Correctly handling large baseline inputs, increasing the navigation range and overcoming limitations caused by slow computations and expensive hardware.
- Handling classical perspective images as well as equirectangular images, both as input and output, supporting many different flavors of 6-DoF applications: point and navigate on a smartphone, stereoscopic immersive viewing on a HMD, light field display, etc.

Applications of the proposed view synthesis algorithm are presented on a HMD device (real-time free navigation), on a light field Super-MultiView 3D Holographic screen (light field rendering) and on holographic stereograms (holography).

II. RELATED WORK

A. ADVANCES IN DEPTH IMAGE-BASED RENDERING

Depth Image-Based Rendering [15], [33] is a technique to render a view from a novel/virtual viewpoint using a number of input images (natural or synthetic content) and their associated depth maps (see Figure 2). This technique lying between Image-Based-Rendering and mesh reconstruction takes advantage of the geometric information in the depth maps while avoiding artifacts due to mesh rendering (Figure 4). It has been successfully used for free navigation in 6-DoF for custom built rigs [41], [42] and 360° video content of static scenes [43]. In essence, to create a virtual view from existing camera views, the pixels in the available views are shifted proportionally with the pixels' disparity. This is the case of simple linear camera setups with parallel optical axes. In general, for an arbitrary camera setup, a reprojection should be applied for each view, as explained in Section III-B.

Most of the DIBR algorithms which are implemented on CPU take several seconds to minutes to render just a single frame [32], [34]. Several GPU implementations [31], [34], [44] have been proposed in an attempt to make DIBR real-time using solely one input view, but limiting the navigation range. Using stereo views increases the possible viewing area and navigation range but the high overlap between the images is still associated with disocclusions. A solution is to add more input views with complementary content. However, increasing the number of views severely increases the computational complexity and impairs attempts to achieve real-time rendering.

Real-time DIBR has already been used in a HMD for mixed reality [45] to warp the filmed background to the eye position, a few centimeters away. ULR [17] renders high quality light-fields in real-time using a geometric proxy instead of the depth maps to correctly blend several input views. Video rendering for non-static scenes has been recently addressed in [22], which enables light field reconstruction from a sparse set of views and omnistereo real-world video rendering on VR devices. However, as stated in [22], the circular camera rigs used for creating omnistereoscopic videos are not suited to the popular and commonly available multiview stereo approaches. Hence, DIBR methods cannot be accommodated for such circular camera rigs, as robust depth estimates are difficult to generate [22]. DIBR video approaches have been explored in [6], but no solution has been provided to render dynamic scenes in real-time without preprocessing. Critical challenges remain unsolved: multi-camera systems are hard to synchronize on a frame basis, making the depth estimation difficult. Moreover, the virtual views corresponding to any head pose need to be rendered in a short time after the depth map generation.

A different approach to address the disocclusions includes inpainting [46], [47] or using superpixels for small

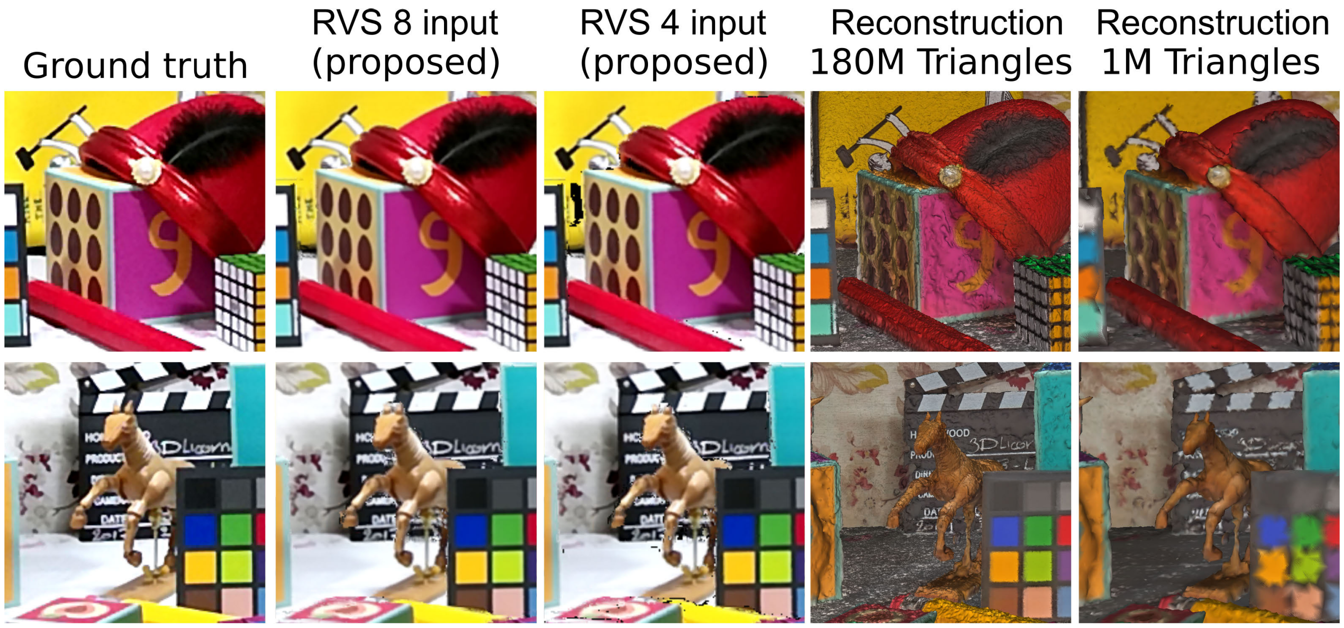


FIGURE 4. Visual comparison between RVS (4 and 8 input views) on zoomed details of Toystable dataset [2] at resolution 1920 × 1080 and mesh reconstruction (180M triangles, and 1M Triangles, decimated) with associated ground truth. The mesh reconstruction was created with the CapturingReality software [39] using around 500 pictures [40] and is rendered using Phong shading.

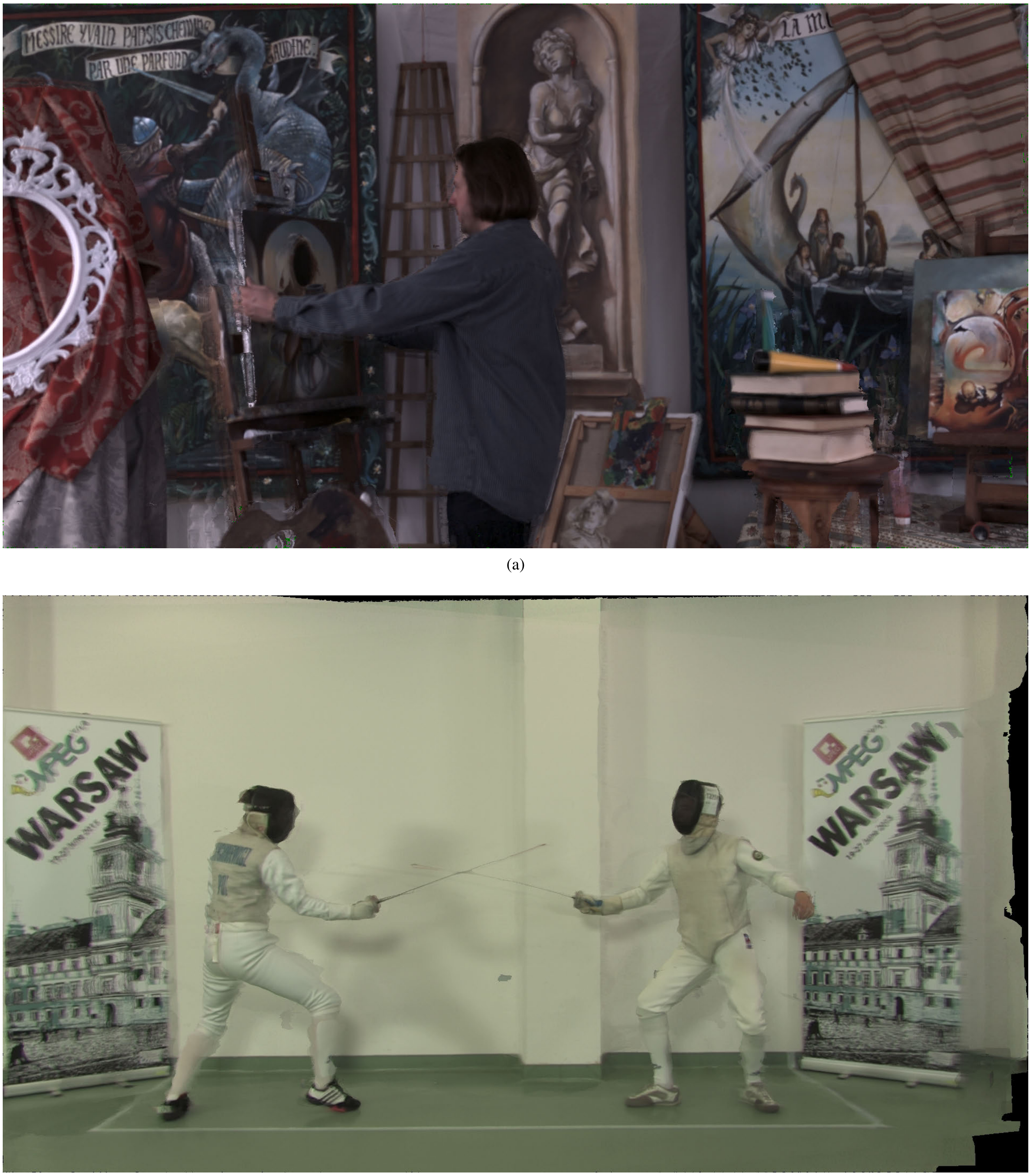
disocclusions [48]. Both techniques suffer from several limitations. On the one hand, inpainting relies on diffusion [49], patch-based methods [50] (traditional) or deep learning [51]. Traditional methods give inconsistent results when employed in video or need long computations [52] while deep learning methods have to be trained on similar video content in order to obtain satisfactory results. On the other hand, superpixels are computationally expensive and improve the results only in very particular cases (when performing navigation toward the scene, also called step-in). For real-time applications, we consider that increasing the number of input views [53], [54] while ensuring that the scene is captured by a large number of cameras from different locations, is the most parallelizable method to limit disocclusions. While the time and memory consumption of using a higher number of input views can be mitigated in addition to the disocclusions issues, the generation of high-quality visual results depends directly on multiple view blending and on the estimated depth maps quality.

Blending synthesized images to obtain the final virtual view may create artifacts due to color and depth inconsistencies, or blurry effects due to calibration issues. The artifacts can be minimized by capturing the scene with controlled light and applying color correction algorithms. In natural scenes, even with Lambertian surfaces and controlled light, color correction is often needed, as the cameras may have color inconsistencies. [55], [56] devised methods to select the most correct color between the views, and robust color correction in multi-camera systems can be obtained for both narrow [57] and large baselines [58].

Regarding depth quality, while depth inconsistencies can be avoided by multi-pass rendering [59], it is preferable to obtain high-quality depth maps as prior. In natural scenes, depth maps cannot be perfectly acquired. When depth maps are obtained using depth sensors, they suffer directly from the sensor's noise, influencing dramatically the severity and number of artifacts in the final rendering. Such depth maps have to be denoised and super-resolved as their spatial resolution is typically lower compared to the ones estimated from RGB sensors [60], e.g. via stereo matching. When the depth maps are computed using multi-stereo matching algorithms [61], the quality is bounded by the algorithm. Color corrected images give more robust results. In our approach, we employ multi-stereo matching algorithms for real scenery, as detailed next.

B. DATASETS AND ASSOCIATED DEPTH MAPS

Throughout this paper, we illustrate our methods using six representative datasets with categorical difficulties to explain our approaches and experimental results. Hence, we introduce datasets and the related tool to acquire their depth maps. The first one is the publicly available MPI-Sintel dataset [62], containing 23 sequences of 50 frames of synthetic videos and their associated depth maps. As synthetic content, the generated depth maps are perfect, but each sequence only has one moving camera. The other datasets are originating from the MPEG-I community [63] that develops compression schemes for immersive video, along with pre- and post-processing tools, like depth estimation and view synthesis [64]. Classroom is a synthetic content rendered from the Blender sample scene classroom in equirectangular projection [4]. Museum [3] is a synthetic composition of green-screen captured persons with a background into equirectangular images covering 180°. The other datasets are natural scenes, with imperfect depth maps. Toystable is a static scene [1], [40] with two sets of depth maps, acquired



(a)

(b)

FIGURE 5. **View synthesis for natural video content with eight input views. (a) Painter, courtesy of InterDigital. (b) Fencing, courtesy of Poznan University of Technology.**

with Kinect v2 and estimated by stereo matching. It contains three camera arrays capturing the scene from three distances: 55cm (Plane A), 85cm (Plane B), 105cm (Plane C). Those datasets are shown in Figure 3. Painter [65] and Fencing [66] are multi-video datasets acquired from camera rigs and are shown in Figure 5.

For the natural datasets, to generate high-quality depth maps, we used MPEG's Depth Estimation Reference

Software (DERS) [67], which allows spatially accurate and temporally coherent depth maps in dynamic scenery.

III. PROPOSED APPROACH

A. OVERVIEW

The proposed method achieves high-quality view synthesis while keeping a low bandwidth for virtual reality applications. To synthesize novel views, we adapt DIBR into the OpenGL pipeline. DIBR consists in remapping an image to a new viewpoint as a function of its pixels' depth. In our design, we deproject the inputs to the 3D space and reproject them to the new viewpoint (Figure 2). The deprojected image forms an implicit mesh, where adjacent pixels are automatically linked (Figure 8). Unfortunately, the faces located on disocclusions are then elongated, which impacts on the quality of the rendered views. One of the key challenges of our approach is to warp and combine the input images to keep only the best part of each of their associated mesh, in real-time.

The important steps of our high-quality view synthesis approach are summarized in Figure 1. Steps 1-4 (acquisition, depth map creation and encoding) are performed offline as a preprocessing stage. Step 5 (loading the images/videos on GPU) is the transmission bottleneck for embedded applications with dynamic content, hence we perform a compression using the GPU hardware before transmitting the information to the decoder and renderer. Steps 6-8 are implemented in this proposed work, necessarily being real-time for a live experience.

We now describe the process to render one image (a frame in a HMD or a view for a light field display). More details about those steps will be given in Sections III-B1 and III-B2.

Each input view is warped following the target camera pose before being blended with the other views (see Figure 6). In the first step, to avoid small holes in the final output image, each triplet of adjacent pixels is linked together to obtain an implicit triangular mesh (see Figure 6). The triangles are then warped to the target virtual view position using the corresponding depth map and are finally rasterized with their associated colors. The pixels detected as lying on disocclusions are discarded and remain black in the final result if no other input image contains information to fill them in (Figure 7).

This method avoids artifacts around abrupt depth variations in the scene (disocclusions), without being as time consuming as inpainting or segmenting the image in superpixels [48], [68]. To detect the triangles lying on disocclusions (or outward facing), a test is performed on a quality criterion q to avoid elongated triangles, empirically defined as follows:

$$q = \begin{cases} 0 & \text{if } L > T - 1 \text{ Pixels or } \vec{n} \cdot \vec{C} \leq 0 \\ T - L & \end{cases} \tag{1}$$

where L is the longest side of the triangle, T a threshold, $\vec{C}$ the target camera viewing vector and $\vec{n}$ the triangle's normal. This quality describes only the size of the triangles. Elongated triangles have then a small quality corresponding to visually unpleasant results. Triangles with zero quality are discarded. Using normalized coordinates results in removing the same pixels at any resolution. However, an artifact visible at one resolution could be ignored at a lower resolution, depending on the user's appreciation. A value of $T = 15$ pixels has been chosen empirically for the Oculus Rift's resolution (1200×1080 per eye) and should be adapted proportionally to the target image's resolution.

For pixels where two triangles are superposed, a test prioritizes the one with the lowest depth $d \in [d_{\min}, d_{\max}]$ given by the input depth map and highest quality q. Empirically we found that choosing the triangle with the maximum weight w, according to:

$$w = \frac{q}{d^3} \tag{2}$$

was satisfying. This depth test prioritises foreground objects and avoids triangles stretched by the disocclusions. The power of 3 on d counterbalances the impact of q as background triangles tend to remain small and thus have higher q values than foreground triangles. If there is no triangle with a weight $w > 0$, the pixel is left black. The weight w is hence based on the depth and the occlusion locations in order to prioritize foreground objects over background ones, and the background over disocclusions, overcoming

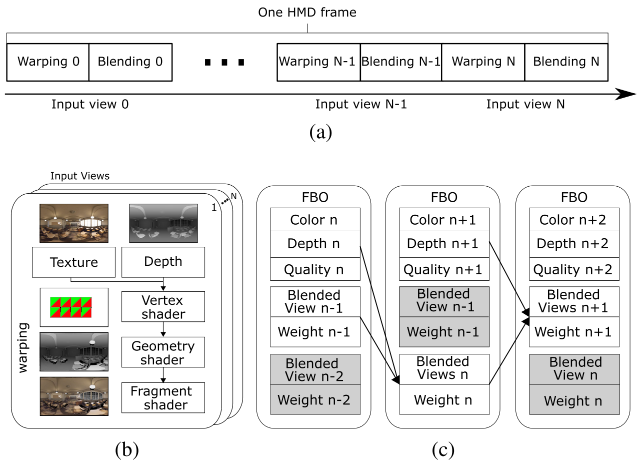


FIGURE 6. Pipeline for rendering one frame. (a) The input images are iteratively warped and blended to render the current frame. (b) Warping of an input view. (c) State of the Framebuffer Object (FBO) at the beginning of a blending phase. The grey cells are the overwritten textures after the blending phase. This double buffering architecture is at the core of real-time rendering.

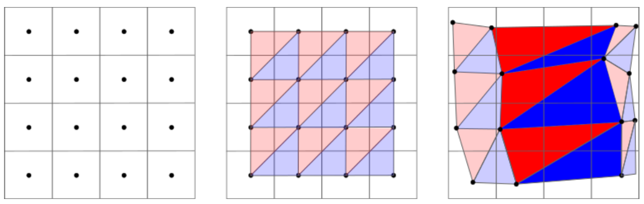


FIGURE 7. Adjacent pixels of an input image (left) are grouped into triangles independently of their depth (center) before being reprojected to their new image location (right). Triangles with a side longer than $T = 15$ pixels are detected as lying on a disocclusion and are discarded (plain color).

visual artifacts. This same weight is used during a second phase, blending the obtained synthetic images from several input views with a weighted mean to get the final result, as further explained in the following section III-B2. Comparisons with other methods are given in Section IV-B.

Finally, a dynamic loader has been designed to load the static content (PNG and YUV) as well as video content on Video Random Access Memory (VRAM). It uses one Framebuffer Object (FBO) as in Figure 6, which avoids unnecessary waste of texture memory in the GPU. The video decoder exploits the graphics card with CUDA to decompress the video stream from H.256 to the rendering pipeline with the corresponding frames, before displaying them in the HMD (more details about this are given in Section III-B3).

B. IMPLEMENTATION

This section describes our algorithm in more details: some preliminary technicalities are presented, followed by the description of our real-time warping and blending shaders, the video decoding and finally the view selection. More specific implementation details can be found in [69]. The goal of our implementation is to render high-resolution stereoscopic views at high frame-rates (i.e. as high as 90 FPS) in order to create a comfortable immersive experience without cyber-sickness.

General DIBR algorithms need pre-computed depth maps associated with color images, as well as the cameras' intrinsic and extrinsic matrices for each input view. This information serves to deproject the pixels of the input views into 3D space before projecting them back to the new virtual viewpoint.

The inputs of our view synthesis approach are perspective or equirectangular color images, including their corresponding depth maps and the input camera parameters (location, rotation, focal length and principal point). All the depth maps are converted from a 8, 10 or 16 bit representation, depending on the dataset, to the range $[d_{min}, d_{max}]$. In principle, the method can take an arbitrary number of input views, but their number is limited by the GPU memory, its bandwidth and power.

Whilst usual techniques mesh the depth maps before reprojection or use splatting on a pixel basis, in this work the adjacent pixels are implicitly meshed by grouping them into triplets, creating triangles in the input images that will be filled by the OpenGL rasterization stage, avoiding any cracks in the image, as shown in Figure 7. Moreover, exploiting the rasterization associated to triangles in the OpenGL fragment shaders yields high accelerations for real-time performance, cf. Sub-section III-B1.

To synthesize one frame in the HMD, we use two shaders per input view, as shown in Figure 6. The first shader warps an input view to the target view, using the depth images. The second shader blends all the resulting warped views together (stored as textures) into the final virtual view by calculating their weighted mean. Those two shaders are detailed in Sections III-B1 and III-B2. The rasterization process (implemented using OpenGL) makes possible to fill in real-time some disocclusions and "cracks" that can appear, especially in step-in navigation scenarios.

1) DIBR WARPING SHADER

In the first shader, which performs the warping, the scene is deprojected and 3D transformed to fit the target view's pose, before being projected back to the target view using the classical OpenGL frustum adapted to the HMD camera matrix, or projected according to the perspective or equirectangular target camera.

The shader receives as input the depth map and an implicit mesh containing $h \times w \times 2$ triangles formed by oriented triplets of adjacent pixels (see Figures 6b and 8). The warping is performed during the vertex pass. The weight w (Equation 2) used later for rasterization and blending is computed during the geometry shader pass. Eventually, the fragment pass performs the rasterization (see Figure 6b).

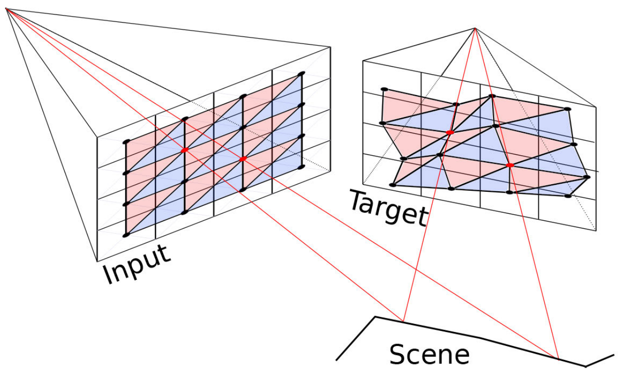


FIGURE 8. Warping shader effect. The triangles formed by three adjacent pixels (left) are deformed in function of the associated depth to fit the new projection (right).

Based on the implicit mesh of the image, we can detect pixels that are to be discarded. Those pixels are the ones lying on disocclusions or with outward facing directions. When a triplet of pixels actually lies on a disocclusion, the triangle becomes very elongated (see Figure 7), creating an artifact. In those cases, they are discarded in the geometry shader pass if one of their sides is longer than T pixels as given in Eq. 1 or the normal is not facing the camera ($q = 0$). After this test, the remaining projected triangles are associated to a quality q for the rasterization and the blending (Equation 1).

In the fragment shader, a depth test is enabled to deal with occlusions. It becomes necessary when two pixels' triplets are warped to the same place, for example when a foreground object occludes the background, due to parallax motion. Similarly to the blending shader of Section III-B2, the chosen color does not only depend on the depth d but also on the quality q to avoid long-sided triangles. The depth test is hence performed on the agglomerated value $\frac{d^3}{q}$ (maximization of Equation 2).

In our implementation, the value of d is normalized for OpenGL's depth test, knowing the highest possible depth d^* for the scene. We have chosen $d^* = 3 \times d_{max}$ as a default

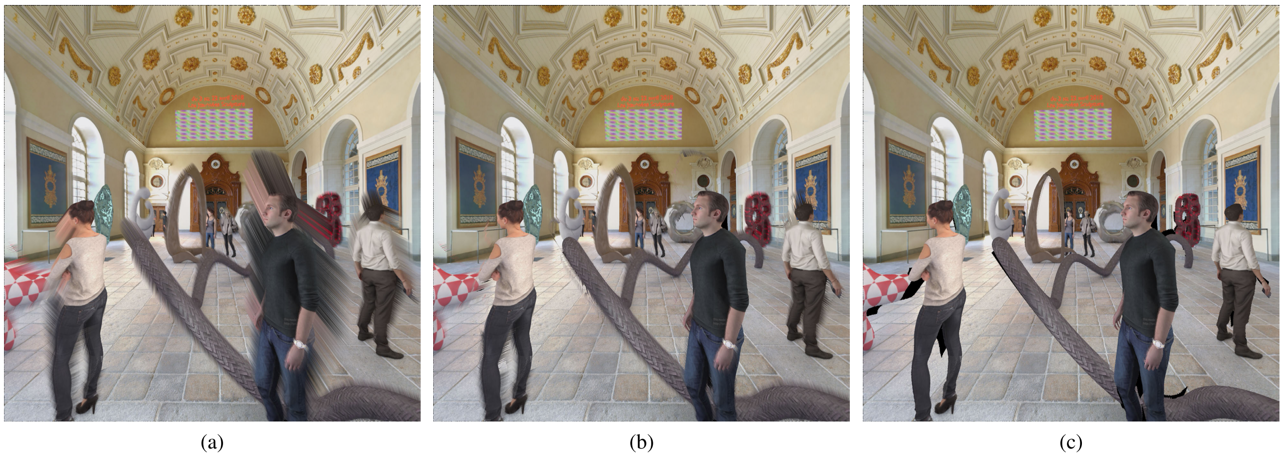


FIGURE 9. **View synthesis from five input views using various qualities and weights. (a) Using only the depth to rasterize and blend the images. (b) Using the quality and depth but not the threshold of T pixels. (c) Using the quality as defined in Equations 1 and 2.**

normalization parameter, where d_{max} is the maximum depth value in the input views, in order to allow reasonable step-out to the user before clipping the scene.

The pixels on disocclusions are thus discarded, preventing a ''halo'' to appear around foreground objects having large depth variations between them and the background (see Figure 9).

2) BLENDING SHADER

The blending shader blends the warped input images together, exploiting a specially designed Framebuffer Object (FBO) (see Figure 6c). This FBO stores the current input view and the previously blended ones as one texture within a ping-pong buffer (grey cells) to limit the memory usage.

The final color of a pixel depends on the shape of the triangles it lies in, and on their associated depths. The following formula prioritizes foreground objects:

$$\mathbf{C} = \frac{1}{W} \sum_{i=1}^{n} w_i \times \mathbf{C_i} \tag{3}$$

where $\mathbf{C}$ is the final color, $\mathbf{C_i}$ the color of image i, w_i the weight of this pixel in image i as defined in Equation 2 and $W = \sum_{i=1}^{n} w_i$ is a normalization factor. Figure 9 shows the impact of using the quality associated with the depth map instead of simply blending the warped images prioritizing foreground objects. One notes that, in contrast to the depth test performed in the warping shader, the choice of the final color is not based on a hard maximum selection, which avoids color and contour artifacts such as described in [23] and [64]. Instead, the color is blended using a weight that prioritizes foreground (low values of d) and small triangles (high values of q), in order to avoid artifacts like those in Figure 9. A soft blending between the warped images can lead to semi-transparency if some of them contain background information. However this effect is attenuated as the foreground is already selected by the hard threshold of the warping pass. Eventually, the final result is transferred to the output display framebuffer.

Blending Quality Analysis: Four kinds of artifacts can be associated with the quality definition. First, temporal inconsistencies in discarding triangles lead to *popping artifacts* (appearing and disappearing triangles). Second, a quality depending on depth map values suffers of quantization: non-uniform quality maps lead to rough blending across the views (*quantization artifacts*). Third, disocclusions occurring in the background are usually less noticeable than large disocclusions in the foreground, due to smaller parallax; discarding triangles of distant disocclusions is worse than keeping the little elongated triangles (*background*). Last, large *step-in* toward an object dilates the triangles, which should nevertheless be kept: discarding it will create holes, transparent or even disappearing objects (step-in artifact).

As explained in the overview, we chose a weight based on a quality computed following Equation 1 (discarding elongated triangles). This empiric quality depends on the target view position relative to the inputs, which may cause popping artifacts when disocclusions or great zooms occur. Nonetheless, with a sufficiently high number of input images (e.g. four), those poor quality pixels are replaced by the information available in other input images. Those artifacts are indeed the most visible in the supplementary videos of MPI-Sintel dataset [62], which uses only one input image.

To overcome those artifacts, different qualities have been explored, based on the depth differences between adjacent pixels, and the ratio over the lengths of the sides of the triangle. Visual results and examples of quality maps are given in Figure 10 while Table 1 provides the advantages and flaws of those methods.

As shown in Figure 10a, a first approach takes into consideration the depth variation between the vertices of the triangles to discard, and is stable in step-in scenarios and immune to popping and depth map quantization artifacts, as the triangles do not depend on the position of the viewer. However, in the background, too many triangles are discarded as the depth map varies abruptly, but motion parallax is sufficiently small to avoid artifacts. Moreover, at the foreground,

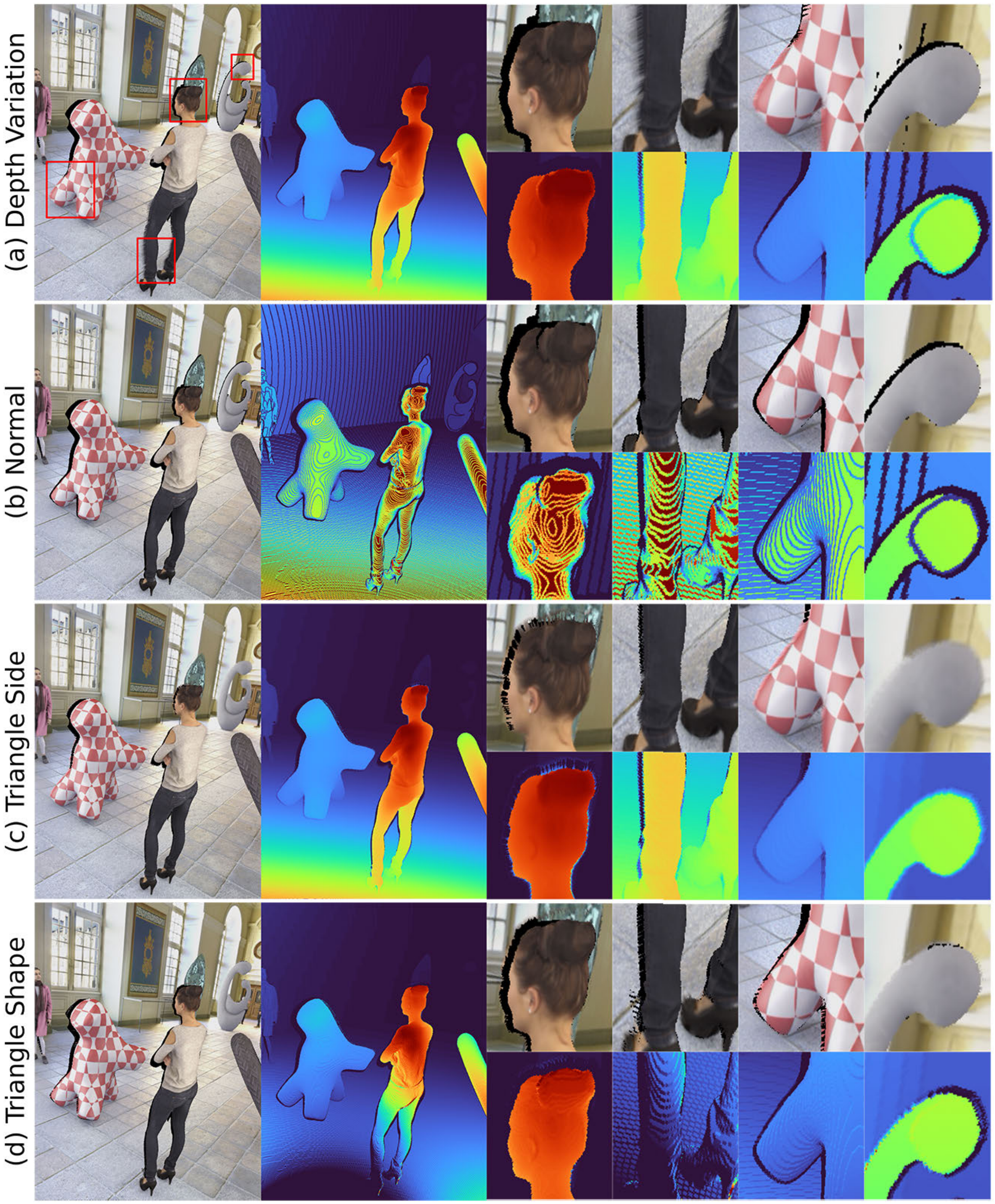


FIGURE 10. Visual comparison of the quality maps for blending with zoomed details in Museum dataset [3]. The color images are blending four warped input views and the displayed quality maps in false colors are extracted from the first input view. (a) Cutting triangles with high depth variations. (b) Cutting triangles facing perpendicularly to the input camera. (c) Cutting triangles with a side larger than T pixels. (d) Cutting triangles with elongated shapes. The head and foreground sculpture illustrate the advantage of a metric detecting disocclusion ((a) and (b)). The background sculpture and legs illustrate the advantage of a metric discarding elongated triangles ((c) and (d)). The details are from close to the camera to far away objects, the last column has colors remapped for visibility.

TABLE 1. Blending coefficients advantages/drawbacks depending on which quality they are based on - Popping: Resistant to popping artifacts in VR (due to target position dependant quality metric), Depth map: not sensitive to depth map quantization, Background: Keeps borders of background objects, Step-in: No dilation artifact in large movements toward the scene. Figure 10 illustrates the effect of each quality metric.

Quality / Artifacts	Depth Variation	Normal	Triangle Side	Triangle Shape
Popping artifacts	✓	✓	✗	✗
Depth map quantization	✓	✗	✓	✓
Background	✗	✗	✓	∼
Step-in	✓	✓	✗	✓

the triangles are not discarded due to the opposite problem. A variable threshold could be devised to control this behavior.

A second approach is to mark triangles which are perpendicular to the input viewing position as disocclusions (cf Figure 10b). This technique is still stable against popping artifacts and step-in deformations as it is independent of the viewpoint and it performs slightly better on foreground and background than the previous one. However, it now suffers from depth map quantization artifacts, visible in Figure 10b. Qualitatively, it is nevertheless a good tradeoff between the techniques.

Discarding triangles with an elongated side in the synthesized view of Figure 10c is resistant to depth map quantization and background issues: the background becomes blurrier without inpairing the realism of the scene. However, popping artifacts appear as the quality maps depend on the viewing position.

Finally, discarding triangles with elongated shape in Figure 10d (high ratio of longest side over shortest side) is similar to the previous method. It performs better in step-in, as zoomed triangles can have a large side while maintaining a regular shape. But, again, small disocclusions with low parallax of the background may be discarded as the corresponding triangles are elongated.

In general, finding the correct quality is a hot topic in the view synthesis field. For instance, LLFF [25] or Free View-Synthesis [70] rely on learning to infer the blending weights. Moreover, at the time of writing, MPEG-I is testing other weights for the blending of the images, based on the rays' direction similarity [64], [71].

In summary, all our proposed qualities have trade-offs and are suited for different navigation types (large step-in or windowed navigation) and scene geometries (depth range). They are all run-time efficient and do not require offline processing, which is an advantage to achieve real-time multi-view synthesis. In the remainder of the paper, comparisons are made on images that are synthesized using equation 1, corresponding to the quality based on the triangle side.

3) REAL-TIME VIDEO DECODING AND VIEW SYNTHESIS

In this section, we discuss the implementation of our approach for HMD supporting navigation with full parallax. Our view synthesizer is able to apply our OpenGL pipeline to a multi-video input. The input videos are decoded to the input textures and depth maps in real-time with CUDA. Beside texture loading at each new video frame, the pipeline stays unchanged for the synthesis part.

Our video content was initially available in YUV format where the frames are not compressed. For example, in the Fencing dataset, 250 frames at 1080p use 741 megabytes for one YUV color video and 988 megabytes for a 16-bits depth video encoded in 400 YUV format. This data volume cannot be transferred to the GPU in real-time due to transfer bandwidth limitations, nor at once due to limited memory. For short videos, the GPU memory can handle uncompressed video frames, but decompression on GPU becomes quickly necessary with longer input (e.g. 4 input video streams of 100 frames with 960 × 540 resolution). While this paper does not focus on compression, we devised several strategies to render video content in real-time.

Before loading the videos, in a preprocessing step, the color and depth content are encoded from YUV420@4k and YUV400@4k to H.265@1080p formats which results in compressed video files (several megabytes), with a compression ratio of about 220 for the color content. This is an important step to be performed, as the amount of data to be transferred to the GPU can be huge.

Our dynamic loader for video content handles several scenarios to optimize the data transfers. This includes multi-pictures (general case, static images), single video with moving camera, short uncompressed multi-video (raw YUV) and multi-video inputs. The various decoding behaviours are shown in Figure 11.

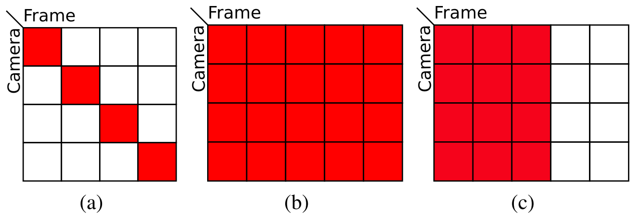


FIGURE 11. Dynamic loading behaviours for video sequences. In red, the frames which are loaded at the beginning of the application. (a) Single video with moving camera: each frame is treated as a different input view, and used when needed. (b) Multi-video (uncompressed): loading all the color and depth frames at the beginning. (c) Multi-video (compressed): loading the first frames for all the views and decompress the following ones when they are needed.

a: MULTI-PICTURES RENDERING (STATIC SCENES)

Static content comes in multi-pictures format. Each input consists of a color image and a depth map with different extrinsic matrices. The pictures are loaded in main memory at the start of the program. For each output frame displayed in the HMD, the corresponding textures are sent one after the other to the FBO for the warping-blending process, as described in Subsections III-B1 and III-B2.

b: SINGLE VIDEO WITH MOVING CAMERA

The MPI-Sintel dataset [62] is a typical example of a scene synthesized from successive viewpoints. If this case is treated

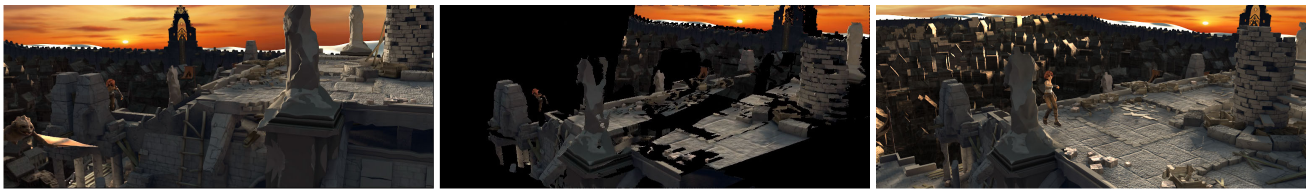


FIGURE 12. Sintel Temple Dataset. Left: input frame (frame 1), middle: synthesized frame 49, right: target frame 49. This shows the importance of view selection among available inputs to minimize dissocclusions artifacts. As the sun is rapidly rising during the movie, we can observe that the houses' roofs don't have the same colors and as this is a movie, the characters moved between frame 1 and frame 49.

as the multi-video case, the user would experience the same motion as the camera without performing the actual movement. However, to avoid motion sickness in the HMD, it is preferable to treat each video frame as a single camera with different extrinsic parameters and leave the user at a static position. For this reason, the input frames are loaded exactly as in multi-picture rendering. During the rendering, each frame is used in the rendering pipeline only when the execution time corresponds to that frame (see Figure 11a). For a movie playing at 30 FPS, and the view synthesis running at 90 FPS, each frame will hence be used three times to synthesize the view corresponding to the user's head position, before going to the next frame. This kind of content leads to many disocclusions (due to synthesis from one single view), which depend on the current camera.

c: MULTI-VIDEO STREAMS

Datasets like Fencing [66] or Painter [65] include several viewpoints on the same dynamic scene, acquired with a synchronized camera rig. However, the VRAM cannot handle multiple video files and their depth without compression (unless the videos consist of only a few seconds at low resolution), see Figure 11b. We improved our view synthesizer in order to load compressed color and depth videos at startup, as explained in Subsection III-B3.

The frames are decompressed during view synthesis each time a new frame is needed (see Figure 11c) and are gradually overwritten when new frames come in. The newly decompressed frames take the place of the oldest textures in the GPU memory, and then are used in the pipeline, as described in Subsections III-B1 and III-B2.

To further optimize the method, we encoded the depth maps as one channel of a RGB video, hence the depth information is contained only in 8 bits. However, the rendering often needs at least 10 bits or even 16 bits to avoid artifacts. In that case, the remaining two channels are used to store more bit-planes from the depth information.

4) VIEW SELECTION

To reach a high frame rate for a better immersive experience, it is recommended to limit the number of input viewpoints. However, to fill as many disocclusions as possible (the other being left in black), it is important to select the k best views among all the available inputs. The choice of k will be discussed in Section IV, to keep a balance between frame rate and visual quality.

This leads us to the problem of selecting the most appropriate views, which is challenging and linked to the choice of the blending weights. The selected views need to be similar to the target (viewing direction, part of the scene captured), but also different between each other, so that each input gives new information about the target view. To find the optimal inputs, a naive idea could be to keep inputs with a small distance to the optical center of the target [64], or with small similarity metrics between each other. However, these measures lack the information about the depth of the view and the cameras' orientations. For example, rendering a view from a quasi-perpendicular camera results in many occlusions even if the optical centers are rather close to each other (see Figure 12).

Due to the low number of input images and the real-time requirements, we avoid using involved view selection methods found in Structure-from-Motion approaches [11], [72]. We propose a solution which computes the frustums intersection between the target and input cameras, and uses this information to obtain better results than with only the distance between the optical centers. To keep the information of the camera direction, we multiply the frustum intersection volume $V(F_1 \cap F_2)$ by the dot product of the two cameras viewing directions, like in the Unstructured Lumigraph [17]. Hence we get a similarity $S_{i,j}$ between the cameras i and j:

$$S_{i,j} = \vec{n}_i \cdot \vec{n}_j \times V(F_i \cap F_j) \tag{4}$$

As each camera can be a classical perspective camera or an equirectangular image (360 degrees or less), the trapezoid can be a generic deformed cube or parts of a sphere delimited by the depth range, which is known in the depth map. There is no simple general formula for those intersections (except sphere-sphere), so the overlapping frustum between two views is estimated using a Monte Carlo method for volume estimation [73]:

$$\hat{V}(F_i \cap F_j) = \frac{V_{BB}}{n} \sum_{k=1}^{n} \mathbb{1}_{p_k \in F_1} \times \mathbb{1}_{p_k \in F_2} \tag{5}$$

where V_{BB} is the volume of a box bounding the two frustums, p_i, $i \in [1, n]$ are randomly drawn points in that bounding box and $\mathbb{1}_{p_k \in F_i}$ is an indicator function evaluating whether p_k belongs to the frustum F_i. Combining Equations 4 and 5, we get an estimated similarity $\hat{S}$ between views i and j with the estimated volume $\hat{V}$:

$$\hat{S}_{i,j} = \vec{n}_i \cdot \vec{n}_j \times \hat{V}(F_i \cap F_j) \tag{6}$$

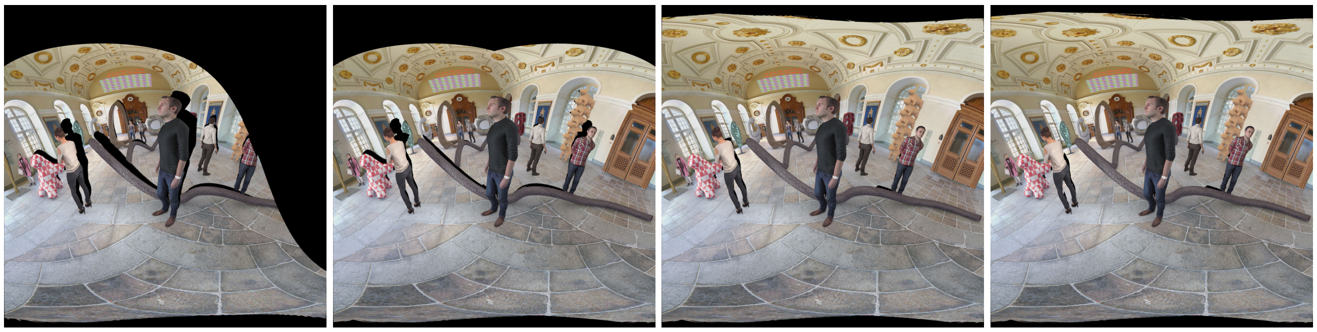


FIGURE 13. Museum Dataset (180° equirectangular images). Result of the view synthesis with view selection for one to four selected input views among eight proposed inputs.

Finally, to render an image using view selection, we select the k views overlapping the most the target camera C according to the metric $\hat{S}_{i,C}$. The set of the k chosen cameras among the N is:

$$\{a_1, \ldots, a_k\} \quad \text{such that } \hat{S}_{a_1,C} \leq \ldots \leq \hat{S}_{a_k,C} \leq \ldots \leq \hat{S}_{a_N,C} \tag{7}$$

This measure still lacks the information about redundant selected views, which could be implemented using the similarity $\hat{S}_{i,j}$ between the inputs views, but does not guarantee that an input view containing unique information will not be discarded. However, the results are satisfying for inputs in a general position and is suitable for real-time applications. Results of the view selection process are shown in Figures 13 and 14.

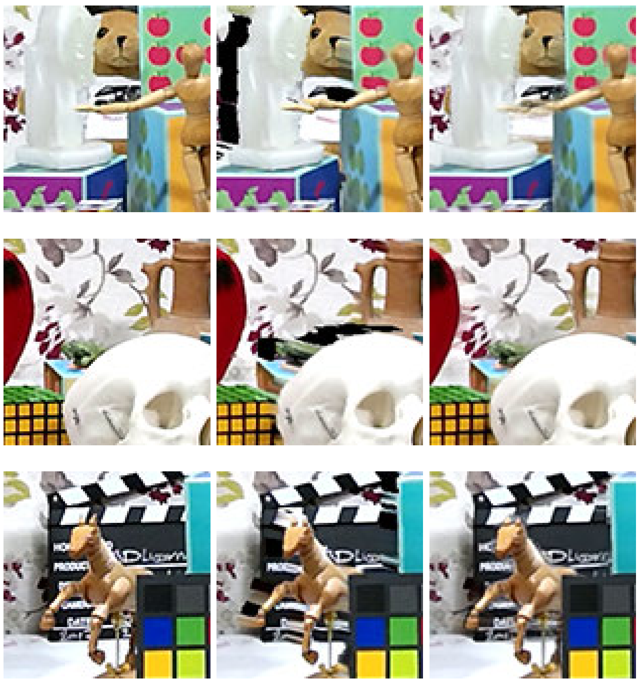


FIGURE 14. Zoomed details on Toystable Dataset. Left: Reference image. Middle: View synthesis from 1 input view. Right: View synthesis from 4 input views. With four input views, very little artifacts and disocclusions remain (mannequin's arm, shadow of the skull on the Rubik's cube, horse's ears and back leg, . . .)

IV. PERFORMANCE EVALUATION

Quantitative evaluations of the frame rate (FPS) and the synthesis quality (PSNR in dB, and MS-SSIM) following natural head movements on an Oculus Rift HMD were done on a low-end as well as a high-end computer; the low-end one is a Linux PC and uses an Intel i3 3225 CPU and a NVIDIA GTX 1660 Super GPU, while the high-end computer is a Windows PC with Intel Xeon E5-2680@2.7GHz CPU and NVIDIA GTX 1080TI GPU.

A. SPEED

Ideally a real-time application should run at least at 90 FPS for VR (and preferably even 120 FPS) [41], [74]–[76], and 30 FPS for light field displays. Video content has usually a frame rate of 30 FPS, which is hence a lower bound for the view synthesis. As we render frames at the eyes' positions at 90 FPS, each input frame is reused three-times. The number of FPS will of course depend on the size of the input images and their number.

To test the speed performance of our software, view synthesis was run in an HMD with one to eight input views on datasets with different resolutions. The content was evaluated on an Oculus Rift HMD, rendering two 1080×1200 images, for a total of 2160×1200 pixels at 90 FPS. The code works with the OpenVR [77] library and is thus compatible with any available headset supporting the OpenVR initiative.

Figure 15 shows the frame rate (two frames are rendered, one for each eye in the HMD) as a function of the number of input views on the high-end computer.

To keep the navigation real-time on our test platform, eight input views with 1920×1080 textures are perfectly manageable. For higher resolutions, the same execution speed can be reached with three to four inputs views.

B. VISUAL QUALITY

To objectively estimate the visual quality, we devised three experiments. In the first experiment we compute the Peak Signal to Noise Ratio (PSNR) and Multi-Scale Structural Similarity Index Measure (MS-SSIM) [78] of our view synthesis against images of Toystable (Plane A), Museum, Fencing and Painter natural datasets for one to eight input views (see Figure 16). As Museum has few overlapping

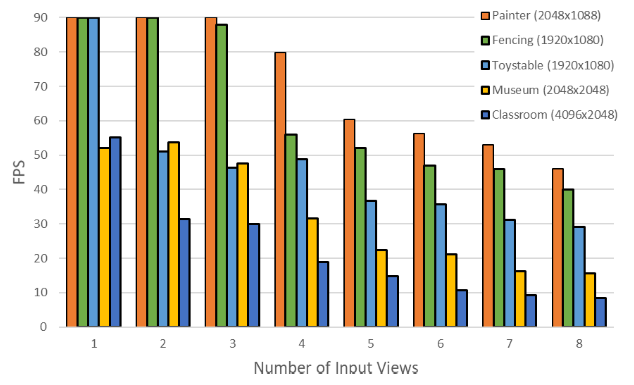


FIGURE 15. Frame rate of the view synthesis on the Windows PC using one to eight input views for Painter, Fencing (video decoder switched on), Toystable, Classroom and Museum datasets. Observe a higher FPS drop with Classroom and Museum due to their input texture sizes.

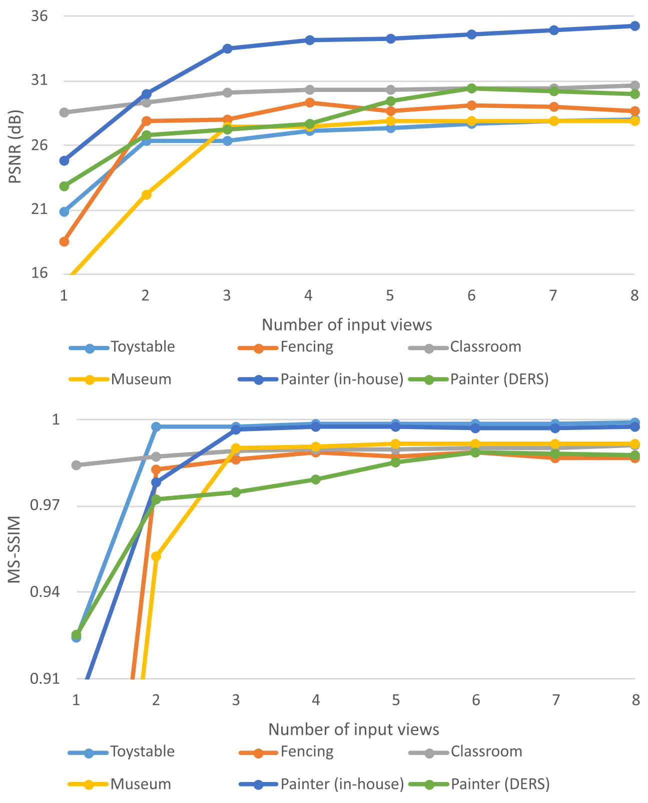


FIGURE 16. Quality trends in PSNR and MS-SSIM in stereoscopic rendering using one to eight input views for Toystable, Fencing, Classroom, Museum, Painter (in-house) and Painter (DERS) datasets.

outward-facing camera viewpoints, the PSNR stops increasing after the four first input views have been selected by our algorithm described in Sub-section III-B4; the four other available input views are looking in the opposite direction and do not bring any new information. Classroom reaches directly high PSNR as the views are 360° equirectangular projections. Intuitively, adding more views improves the quality as more occlusions are covered. However, the graphs in Figure 16 show that three or four images are sufficient to reach a high-quality output for all datasets. Adding more views does not significantly improve the quality but makes

TABLE 2. PSNR, MS-SSIM [78] and IV-PSNR [82] for Toystable dataset at various resolutions. RVS (8) uses eight input color images, provided with their depth maps computed with 6 additional images (14 in total). RVS (4) uses only four input color images, with their depth maps computed with 6 additional images (10 in total) [83], [84]. NERF and LLFF use the same 8 input images as RVS (8) and the same calibration parameters. The evaluations are the average performed on 15 images, including the input poses, on the Toystable (Plane B) dataset.

	PSNR (dB)			
	RVS (8)	RVS (4)	LLFF [25]	NeRF [29]
1920 × 1080	**26.43**	24.67	25.98	25.52372
960 × 540	27.06	25.08	25.68	**28.33**
480 × 270	23.85	22.67	22.13	**28.94**
	MS-SSIM [78]			
	RVS (8)	RVS (4)	LLFF [25]	NeRF [29]
1920 × 1080	0.9979	0.9953	**0.9985**	0.9984
960 × 540	0.9987	0.9958	0.9989	**0.9994**
480 × 270	0.9980	0.9950	0.9980	**0.9988**
	IV-PSNR [82]			
	RVS (8)	RVS (4)	LLFF [25]	NeRF [29]
1920 × 1080	37.47	34.83	37.57	**38.68**
960 × 540	37.94	38.68	37.26	**40.13**
480 × 270	35.00	33.59	33.70	**40.85742**

the frame rate drop noticeably. Therefore, it is probably wiser to add dedicated inpainting extensions to tackle the remaining quality imperfections, e.g. Deep Neural Networks (DNN), cf. Section V.

To emphasize the importance of the depth map on the final quality, an in-house multi-stereo algorithm [79]–[81] was used on a single frame of Painter. Our depth estimation method substantially outperformed DERS on static content, showing an increase in PSNR value up to 5 dB (Figure 16). However, it cannot be used for dynamic content, as the absence of the temporal filtering in the algorithm leads to visual artifacts. Therefore, DERS was used during all our experiments for a fair comparison.

The visual results shown for Museum and Toystable in Figures 13 and 14 respectively, follow the same trend, confirming that with four input views, most of the disocclusion artifacts have disappeared. Moreover, the results shown in Figure 15 confirm that four input views are sufficient to sustain a fluent visual experience with common head movements. Other examples of the view synthesis are shown in Figures 3 and 5. In Figure 4, we subjectively compare the results given by our DIBR technique to a real-time Phong shading performed on a photogrammetry of 180M triangles and its decimation to 1M triangles. We observe that our view synthesis with 8 input views and 4 viewpoints both creates more realistic renderings than mesh based rendering. As such a rendering method does not reach photorealism, we also compare our results with slower algorithms.

In the second experiment, we compare our approach with Mildenhall's promising Local Light Field Fusion (LLFF) [25] and Neural Radiance Field (NeRF) [29] techniques on the Toystable (Plane B) dataset. While LLFF does not need training, NeRF learns a representation of the scene rendered using a *raytracing* approach in volume rendering [85]. It was trained for more than 500k epochs (only 200k in the original paper), cf. Supplementary material. Table 2 shows the

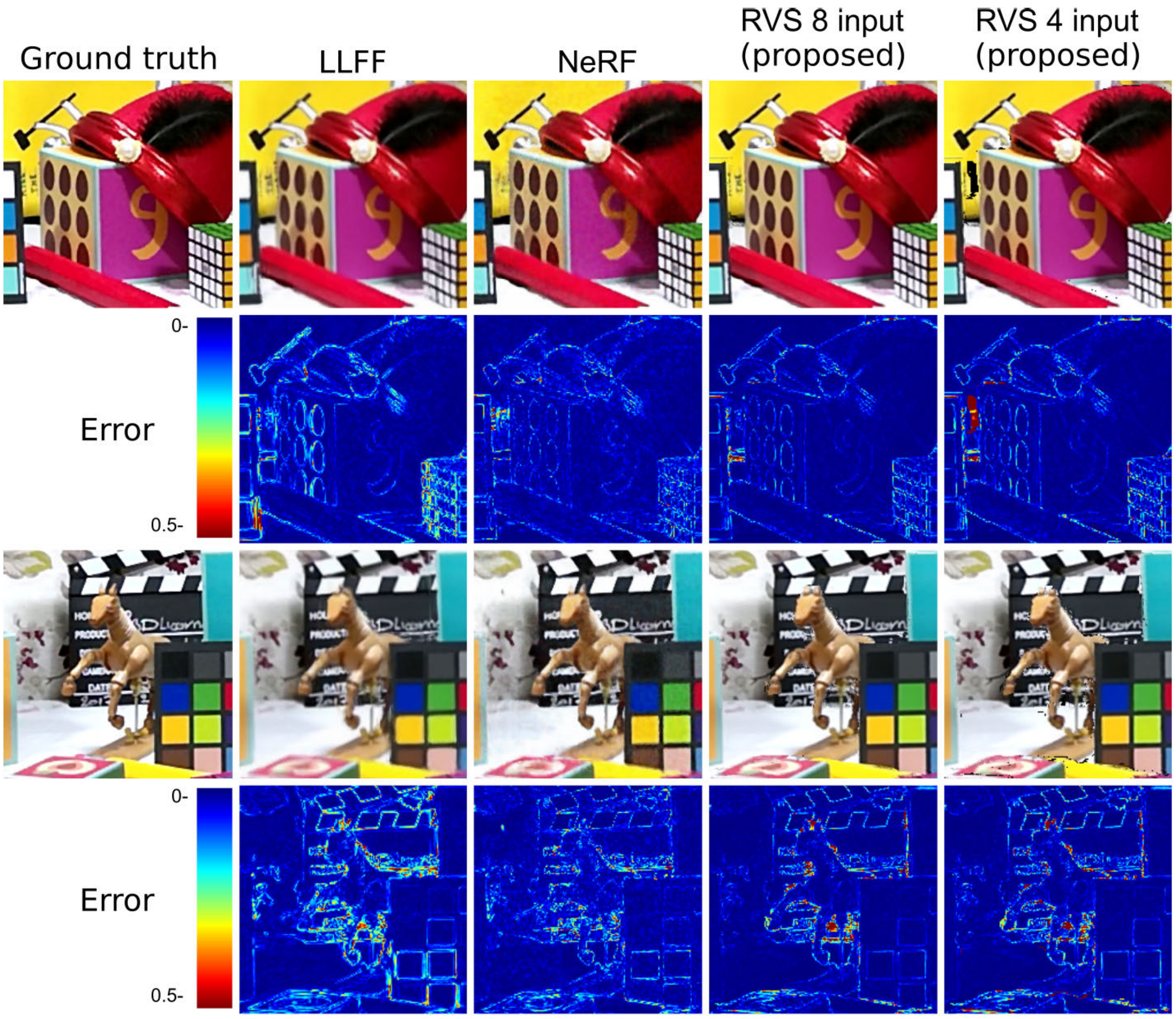


FIGURE 17. Visual comparison between LLFF, NeRF and RVS on zoomed details of Toystable dataset at resolution 1920 × 1080 with associated ground truth difference map. We observe a uniform grainy texture on the error pictures in NeRF and LLFF introduced by the raytracing approach and the neural blending of several MPIs. The occlusions in RVS with 4 input views are removed using 8 viewpoints. Our approach produces visually smoother flat areas and less errors on the edges while suffering more from occlusions.

comparison in PSNR, MS-SSIM [78] and IV-PSNR [82], while Figure 17 shows the comparative results on two challenging details of the dataset. We observe that NeRF performs better on lower resolution images and therefore ran the experiment on three resolutions: 1920 × 1080, 960 × 540, 480 × 270. In full-HD, our approach performs equally well (within 1 dB PSNR) to NeRF and LLFF and renders the views in real-time, while several seconds per frame are needed for LLFF and minutes for NeRF. In lower resolutions, NeRF achieves the best results, but those resolutions and NeRF's computation time are not recommended in practice for VR applications.

In the error plots (difference between each technique and the Ground Truth (GT)) of Figure 17, we observe that neural techniques introduce noise in flat areas. This effect can be reduced in NeRF by using more rays per pixel, but it drastically increases the training time and the network does not fit on our GPU anymore. Another visible difference is that our approach has noticeably less errors on the edges, but it suffers more from disocclusions - the areas that are left black in Figure 17 (white in error).

We also reproduced the tests of [34], describing a GPU implementation of DIBR. The tests consist in synthesizing the 49 frames of each sequence in the MPI-Sintel dataset [62] using only the first frame or only the last frame. Finally, we evaluated the PSNR and MS-SSIM [78] of the resulting images. We followed the original methodology of [34] for evaluation: the images are compared to the ground truth after masking the disocclusions and moving characters. Figure 18 shows a comparison between our results and

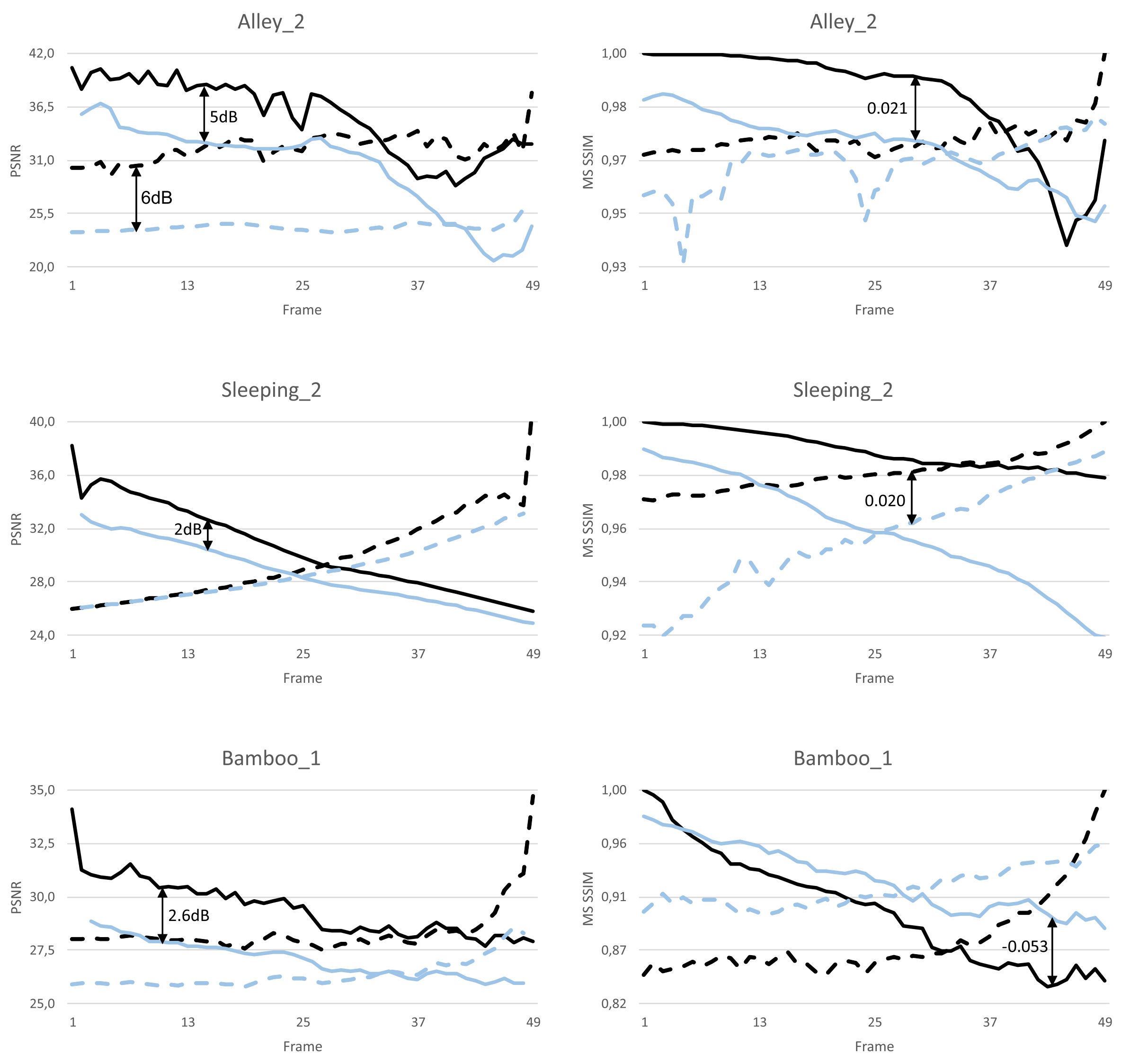


FIGURE 18. **PSNR and MS-SSIM results of view synthesis on three representative MPI-Sintel sequences over frames 1 to 49. Plain lines represent the view synthesis from image 1, dashed lines represent the view synthesis from image 49. Black: proposed, light blue: Ogniewski's results [34].**

Ogniewski's [34]. On average we perform 2 dB (PSNR) or 0.02 (MS-SSIM) better for all test sequences, with a maximum quality improvement of up to 7 dB in PSNR and 0.05 for MS-SSIM. Visual results are shown in Figure 19, showing the high-quality outcome (except for the disocclusions coming from one camera synthesis).

Hence, our method obtains the highest quality while maintaining real-time performance with 4 input images.

C. APPLICATIONS

Previous sections have focused on a typical HMD-VR application where any virtual viewpoint is synthesized from a couple of fixed camera feeds. To sustain stereoscopic HMD viewing, two virtual views (one per eye) must be synthesized. We present now three different applications where our approach can equally well perform: (a) Light field displays (e.g. Holografika display) that projects hundreds of parallax-consistent views of the scene to the user. This process requires more than two views to be simultaneously synthesized.

If all these views can be synthesized in the display from a limited number of input views, huge gains in video streaming data rates can be expected. The same line of thought can be used for (b) holographic stereograms where any number of views can be synthesized to produce a high-quality display. As our approach is robust in interpolation and extrapolation, we reached high-quality results in this context as described hereafter. We have also developed a website (c) to dynamically show our results and let third parties experiment with our software.

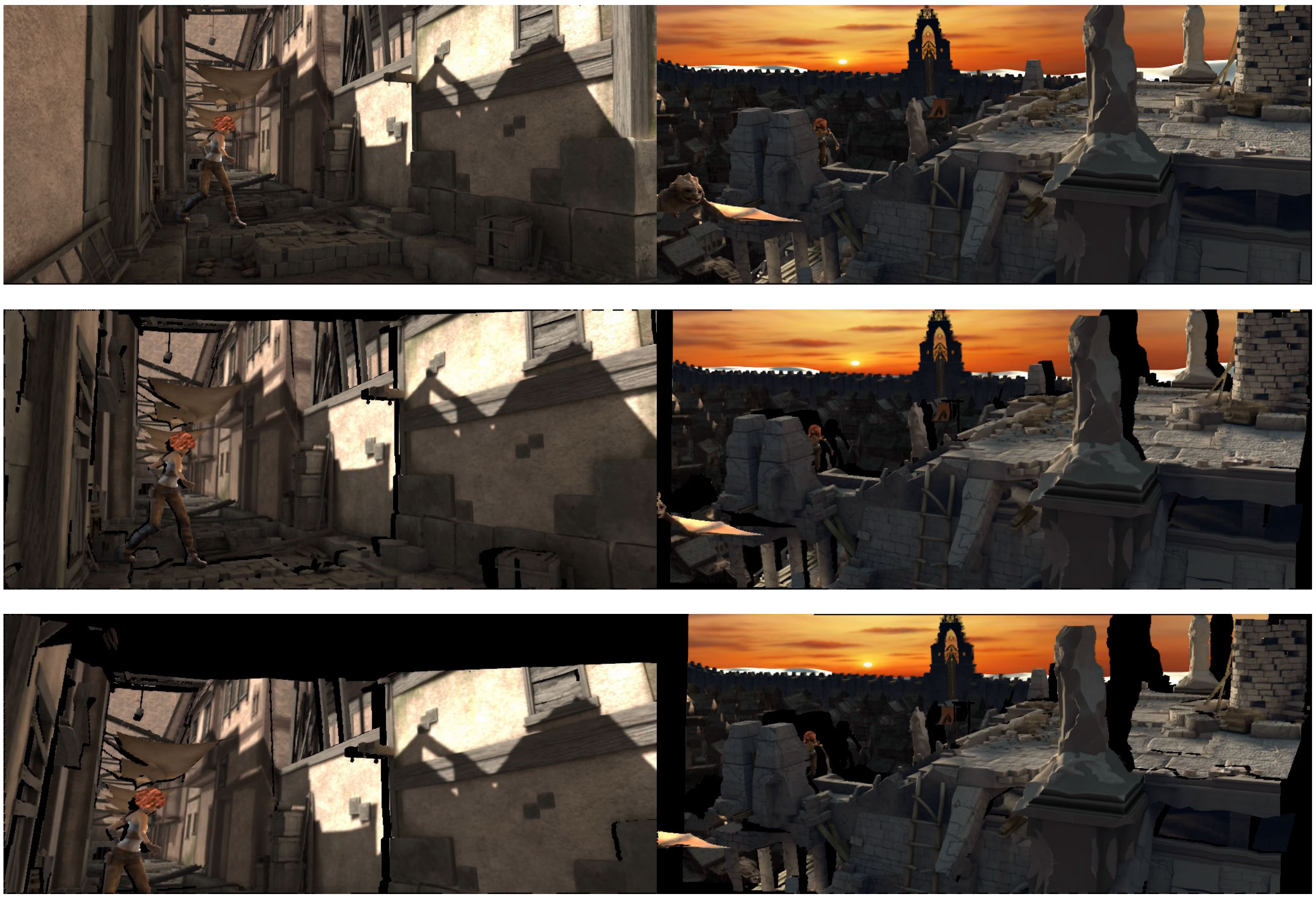


FIGURE 19. Representative graphical results for MPI-Sintel synthesized from frame 1 as input to several selected frames. As the camera is moving and the views are synthesized using only the first frame, occlusions (in black) can be observed. The camera of the second sequence moves sharply, which induces more disocclusions.

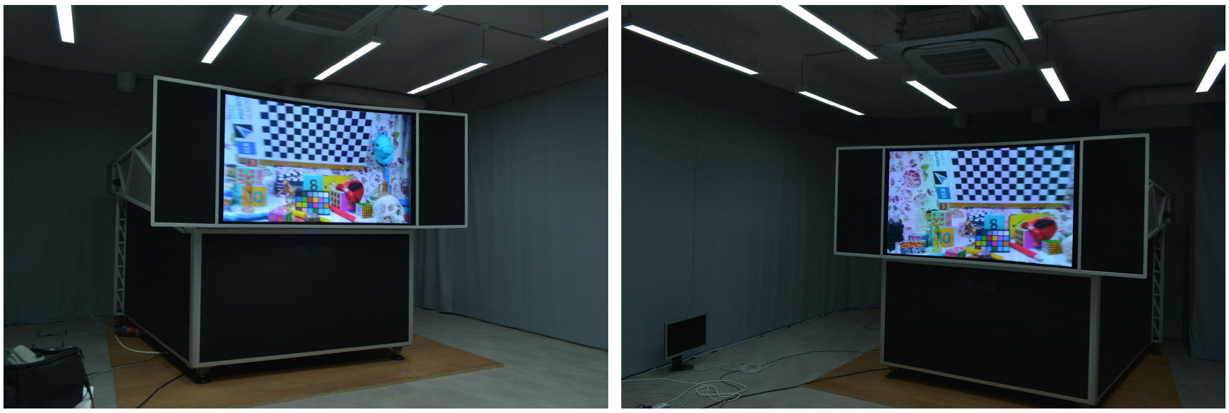


FIGURE 20. Rendering Toystable (Plane A) on light field display (Holografika). 72 viewpoints have been synthesized and fed to the display to simulate horizontal parallax.

1) HOLOGRAFIKA

Holografika is a Super-MultiView light field display [7] screen which uses 72 inlined projectors to render a scene from any viewpoint. We synthesized the video Fencing dataset using only 4 cameras to render 72 videos of 250 frames simultaneously in 720p - the maximum resolution of the projectors - (7s for 72 output views including encoding in YUV420 format on the disk) and the Toystable dataset for a static scene before displaying them on the screen [86]. We obtained high-quality results for the two datasets. In the navigation of the Toystable dataset we can see the objects in 3D from several viewpoints without glasses

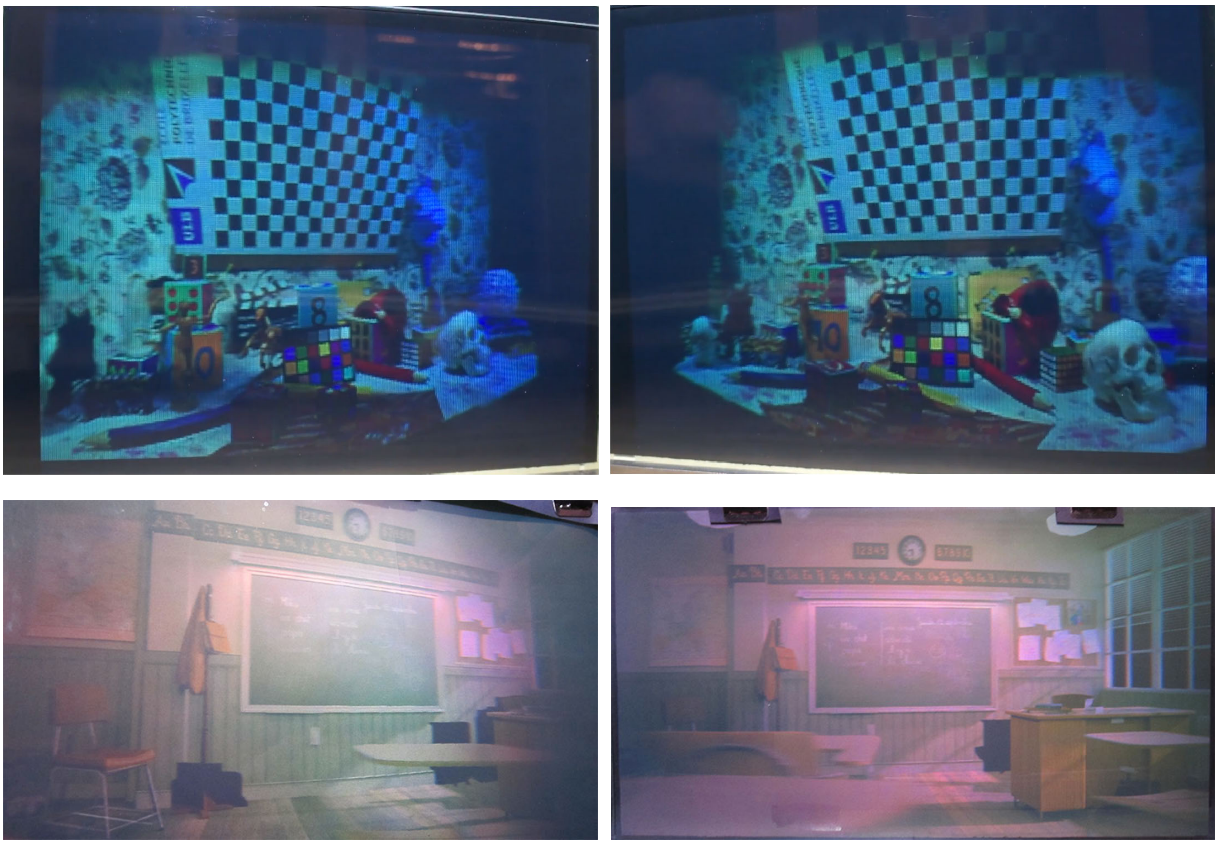


FIGURE 21. Viewpoints of the holographic stereograms. Top: Toystable dataset (200 synthesized views from 4 input images, hogel size of 500μm). Bottom: Classroom dataset (768 synthesized views from 4 input images, hogel size of 250μm). Videos of the holographic stereograms showing the horizontal parallax can be found in the supplementary material.

(Figure 20). Next-generation holographic screens display more than 72 viewpoints. However, the bandwidth requirement to transfer viewpoints at 30 fps is unachievable. This proof-of-concept shows that it is possible to transfer only key images and synthesize the others viewpoints in real-time.

2) HOLOGRAPHIC STEREOGRAMS

Holographic Stereograms are like a light field display, but are implemented differently: the hundreds of projection directions are obtained with diffractive, Holographic Optical Elements (HOE) engraving interference fringes.

Very fine details can be engraved as the discretization of the light-field depends only on laser limitations. The Toystable and the Classroom datasets were used to generate holographic stereograms with respective sizes of 26cm $\times$ 21cm and 40cm $\times$ 22cm (Figure 21). They were engraved using 200 and 768 virtual views with 500μm and 250μm hogel sizes (HOE), respectively synthesized using 4 and 8 input views [87]. A full description of the procedure to make holographic stereograms using our software can be found in [88]. Our solution's quality can be further improved on this application as it is not bound to the real-time component.

3) WEBSITE

A demonstration website https://lisaserver.ulb.ac.be/rvs/ was developed to show the quality of the synthesized images. At the time of writing, the website is limited to a non real-time version of our software to focus on the quality aspects without cumbersome HMD interfacing (no cloud computing application with client VR).

V. CONCLUSION AND FUTURE WORK

Real-time rendering of natural scenes on HMD with 6-DoF is achieved with our approach on commodity GPU platforms using DIBR and proprietary shaders. They allow to reach 90 FPS without dropping quality and without particular optimization or preprocessing, apart from the OpenGL pipeline and depth map priors. One of the main advantages of the proposed approach is the low number of views needed for high-quality synthesis, as shown on the MPI-Sintel (cartoon) sequences (Figure 19), as well as the quality we obtain with natural and synthetic datasets. Our system handles any pose by relying on efficient systems in the selection and the blending, while managing the throughput at each step.

We performed extended experiments with only four input views to render a scene on a light field display with horizontal motion parallax and on holographic stereograms, proving the versalitity of our approach.

We plan to explore real-time DNN inpainting [89], [90] and compression mechanisms for data transmission in embedded VR without impeding on the quality and processing speed. As our method leaves only small disocclusions, DNN inpainting is promising to fill realistically the discarded triangles for light field content. Combining inpainting with compression allows to lower the number of input views even further. We expect new challenges with inpainting, but, we believe that our approach is flexible enough to be adapted to this use case.

SUPPLEMENTARY MATERIAL

Supplementary video and data material is available with our paper. It shows real-time large 6-DoF movements with the same quality as in Figure 14 for various datasets, as well as our *quality experiments* videos, associated to Figure 18. Training data and high resolution images of our blending weight (Figure 10) are also included. Finally, Table 2 is extended to include more details.

ACKNOWLEDGMENT

Sarah Fachada is a Research Fellow of the Fonds de la Recherche Scientifique - FNRS, Belgium. This work uses several copyrighted materials, including Toystable dataset [1], [2] created in the 3DLicorneA project, supported by Innoviris, the Brussels Institute for Research and Innovation Belgium, under contract No.: 2015 R39c, 3DLicorneA. Technicolor Museum and Technicolor Painter, Technicolor. All rights reserved Copyright© 2017-2018-Thomson Licensing [3], and *Classroom dataset* from Philips [4].

REFERENCES

[1] D. Bonatto, A. Schenkel, T. Lenertz, Y. Li, and G. Lafruit, *ULB High Density 2D/3D Camera Array Data set, Version 2*, document ISO/IEC JTC1/SC29/WG11, 2017.

[2] D. Bonatto, S. Fachada, and G. Lafruit, "ULB ToysTable," Tech. Rep., Jun. 2021, doi: 10.5281/zenodo.5055543.

[3] R. Doré, G. Briand, and T. Tapie, *Technicolor 3DoF+ Test Materials [M42349]*, document ISO/IEC JTC1/SC29/WG11, Apr. 2018, p. 8.

[4] B. Kroon, *3DoF+ Test Sequence ClassroomVideo [M42415]*, document ISO/IEC JTC1/SC29/WG11, Apr. 2018.

[5] S. E. Chen, "QuickTime VR: An image-based approach to virtual environment navigation," in *Proc. 22nd Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, 1995, pp. 29–38. [Online]. Available: https://portal.acm.org/citation.cfm?doid=218380.218395

[6] T. Tanimoto, M. P. Tehrani, T. Fujii, and T. Yendo, "FTV for 3-D spatial communication," *Proc. IEEE*, vol. 100, no. 4, pp. 905–917, Apr. 2012, doi: 10.1109/JPROC.2011.2182101.

[7] T. Balogh, T. Forgács, T. Agács, O. Balet, E. Bouvier, F. Bettio, E. Gobbetti, and G. Zanetti, "A scalable hardware and software system for the holographic display of interactive graphics applications," *Eurographics*, pp. 109–112, Jan. 2005. [Online]. Available: https://www.researchgate.net/publication/230735224_A_scalable_hardware_and_software_system_for_the_holographic_display_of_interactive_graphics_applications, doi: 10.2312/egs.20051036.

[8] R. C. Bolles, H. H. Baker, and D. H. Marimont, "Epipolar-plane image analysis: An approach to determining structure from motion," *Int. J. Comput. Vis.*, vol. 1, no. 1, pp. 7–55, 1987, doi: 10.1007/BF00128525.

[9] R. Mohr, L. Quan, and F. Veillon, "Relative 3D reconstruction using multiple uncalibrated images," *Int. J. Robot. Res.*, vol. 14, no. 6, pp. 619–632, Dec. 1995, doi: 10.1177/027836499501400607.

[10] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski, "Building Rome in a day," *Commun. ACM*, vol. 54, no. 10, pp. 105–112, Oct. 2011. [Online]. Available: https://dl.acm.org/citation.cfm?doid=2001269.2001293

[11] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proc. IEEE CVPR*, Jun. 2016, pp. 4104–4113. [Online]. Available: https://ieeexplore.ieee.org/document/7780814/

[12] M. Botsch, M. Spernat, and L. Kobbelt, "Phong splatting," in *Proc. 1st Eurograph. Conf. Point-Based Graph.*, 2004, pp. 25–32, doi: 10.5555/2386332.2386338.

[13] D. Bonatto, S. Rogge, A. Schenkel, R. Ercek, and G. Lafruit, "Explorations for real-time point cloud rendering of natural scenes in virtual reality," in *Proc. Int. Conf. 3D Imag. (IC3D)*, Dec. 2016, pp. 1–7, doi: 10.1109/IC3D.2016.7823453.

[14] M. Levoy and P. Hanrahan, "Light field rendering," in *Proc. 23rd Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, 1996, pp. 31–42, doi: 10.1145/237170.237199.

[15] C. Fehn, "Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV," *Proc. SPIE*, vol. 5291, pp. 93–105, May 2004, doi: 10.1117/12.524762.

[16] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The lumigraph," in *Proc. 23rd Annu. Conf. Comput. Graph. Interact. Techn.*, 1996, pp. 43–54, doi: 10.1145/237170.237200.

[17] C. Buehler, M. Bosse, L. Mcmillan, S. Gortler, and M. Cohen, "Unstructured lumigraph rendering," in *Proc. 28th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, 2001, pp. 425–432. [Online]. Available: http://portal.acm.org/citation.cfm?doid=383259.383309

[18] A. Isaksen, L. McMillan, and S. J. Gortler, "Dynamically reparameterized light fields," in *Proc. 27th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, 2000, pp. 297–306. [Online]. Available: https://portal.acm.org/citation.cfm?doid=344779.344929

[19] G. Riegler and V. Koltun, "Stable view synthesis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2021, pp. 12216–12225.

[20] J. Flynn, I. Neulander, J. Philbin, and N. Snavely, "DeepStereo: Learning to predict new views from the world's imagery," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5515–5524, doi: 10.1109/CVPR.2016.595.

[21] R. S. Overbeck, D. Erickson, D. Evangelakos, and P. Debevec, "Welcome to light fields," in *Proc. ACM SIGGRAPH Virtual, Augmented, Mixed Reality*, Aug. 2018, p. 32, doi: 10.1145/3226552.3226557.

[22] C. Schroers, J.-C. Bazin, and A. Sorkine-Hornung, "An omnistereoscopic video pipeline for capture and display of real-world VR," *ACM Trans. Graph.*, vol. 37, no. 3, pp. 1–13, Aug. 2018. [Online]. Available: https://dl.acm.org/citation.cfm?doid=3243123.3225150

[23] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely, "Stereo magnification: Learning view synthesis using multiplane images," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 1–12, Aug. 2018, doi: 10.1145/3197517.3201323.

[24] J. Flynn, M. Broxton, P. Debevec, M. DuVall, G. Fyffe, R. Overbeck, N. Snavely, and R. Tucker, "Deepview: View synthesis with learned gradient descent," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2362–2371, doi: 10.1109/CVPR.2019.00247.

[25] 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 Trans. Graph.*, vol. 38, no. 4, pp. 1–14, Jul. 2019, doi: 10.1145/3306346.3322980.

[26] M. Broxton, J. Flynn, R. Overbeck, D. Erickson, P. Hedman, M. Duvall, J. Dourgarian, J. Busch, M. Whalen, and P. Debevec, "Immersive light field video with a layered mesh representation," *ACM Trans. Graph.*, vol. 39, no. 4, Jul. 2020, doi: 10.1145/3386569.3392485.

[27] P. Hedman, T. Ritschel, G. Drettakis, and G. Brostow, "Scalable inside-out image-based rendering," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 1–11, Nov. 2016, doi: 10.1145/2980179.2982420.

[28] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, and G. Brostow, "Deep blending for free-viewpoint image-based rendering," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 1–15, Jan. 2019, doi: 10.1145/3272127.3275084.

[29] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2020, pp. 405–421, doi: 10.1007/978-3-030-58452-8_24.

[30] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt, "Neural sparse voxel fields," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 15651–15663, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/b4b758962f17808746e9bb832a6fa4b8-Paper.pdf

[31] B. Reinert, J. Kopf, T. Ritschel, E. Cuervo, D. Chu, and H.-P. Seidel, "Proxy-guided image-based rendering for mobile devices," *Comput. Graph. Forum*, vol. 35, no. 7, pp. 353–362, Oct. 2016, doi: 10.1111/cgf.13032.

[32] O. Stankiewicz, K. Wegner, M. Tanimoto, and M. Domanski, *Enhanced View Synthesis Reference Software (VSRS) for Free-Viewpoint Television [M31520]*, document ISO/IEC JTC1/SC29/WG11, Jan. 2013.

[33] W. Sun, L. Xu, O. C. Au, S. H. Chui, and C. W. Kwok, ''An overview of free view-point depth-image-based rendering (DIBR),'' in *Proc. APSIPA Annu. Summit Conf.*, 2010, pp. 1023–1030. [Online]. Available: https://hdl.handle.net/1783.1/47652
[34] J. Ogniewski, ''High-quality real-time depth-image-based-rendering,'' in *Proc. SIGRAD*, vol. 143, Aug. 2017, pp. 1–8.
[35] J. Jylänki, *A Thousand Ways to Pack Bin—A Practical Approach to Two-Dimensional Rectangle Bin Packing*, 2010. [Online]. Available: http://clb.demon.fi/files/RectangleBinPack.pdf
[36] G. Lafruit, D. Bonatto, C. Tulvan, M. Preda, and L. Yu, ''Understanding MPEG-I coding standardization in immersive VR/AR applications,'' *SMPTE Motion Imag. J.*, vol. 128, no. 10, pp. 33–39, Nov. 2019, doi: 10.5594/JMI.2019.2941362.
[37] S. Fachada, B. Kroon, D. Bonatto, B. Sonneveldt, and G. Lafruit, *Reference View Synthesizer (RVS) 2.0 Manual, MPEG2018/N17759*, document ISO/IEC JTC1/SC29/WG11, Jul. 2018.
[38] J. Kessenich, D. Baldwin, and R. Rost, ''The OpenGL shading language, version 4.60.7,'' Khronos Group, Beaverton, OR, USA, Tech. Rep., 2019.
[39] *Capturing Reality*. Accessed: Sep. 10, 2018. [Online]. Available: https://www.capturingreality.com
[40] A. Schenkel, D. Bonatto, S. Fachada, H.-L. Guillaume, and G. Lafruit, ''Natural scenes datasets for exploration in 6DOF navigation,'' in *Proc. Int. Conf. 3D Immersion (IC3D)*, Dec. 2018, pp. 1–8, doi: 10.1109/IC3D.2018.8657865.
[41] R. S. Overbeck, D. Erickson, D. Evangelakos, M. Pharr, and P. Debevec, ''A system for acquiring, processing, and rendering panoramic light field stills for virtual reality,'' *ACM Trans. Graph.*, vol. 37, no. 6, pp. 1–15, Jan. 2019, doi: 10.1145/3272127.3275031.
[42] A. P. Pozo, M. Toksvig, T. F. Schrager, J. Hsu, U. Mathur, A. Sorkine-Hornung, R. Szeliski, and B. Cabral, ''An integrated 6DoF video camera and system design,'' *ACM Trans. Graph.*, vol. 38, no. 6, pp. 1–16, Nov. 2019, doi: 10.1145/3355089.3356555.
[43] J. Huang, Z. Chen, D. Ceylan, and H. Jin, ''6-DOF VR videos with a single 360-camera,'' in *Proc. IEEE Virtual Reality (VR)*, Mar. 2017, pp. 37–44, doi: 10.1109/VR.2017.7892229.
[44] R. G. de Albuquerque Azevedo, F. Ismério, A. B. Raposo, and L. F. G. Soares, ''Real-time depth-image-based rendering for 3DTV using OpenCL,'' *Adv. Visual Comput.*, vol. 8887, pp. 97–106, Dec. 2014, doi: 10.1007/978-3-319-14249-4_10.
[45] C.-J. Lai, P.-H. Han, H.-L. Wang, and Y.-P. Hung, ''Exploring manipulation behavior on video see-through head-mounted display with view interpolation,'' in *Proc. Asian Conf. Comput. Vis. (ACCV)*, vol. 10118, 2017, pp. 258–270, doi: 10.1007/978-3-319-54526-4_20.
[46] I. Daribo and B. Pesquet-Popescu, ''Depth-aided image inpainting for novel view synthesis,'' in *Proc. IEEE Int. Workshop Multimedia Signal Process.*, Oct. 2010, pp. 167–170, doi: 10.1109/MMSP.2010.5662013.
[47] B. Ceulemans, S.-P. Lu, G. Lafruit, and A. Munteanu, ''Robust multiview synthesis for wide-baseline camera arrays,'' *IEEE Trans. Multimedia*, vol. 20, no. 9, pp. 2235–2248, Sep. 2018, doi: 10.1109/TMM.2018.2802646.
[48] M. P. Tehrani, T. Tezuka, K. Suzuki, K. Takahashi, and T. Fujii, ''Free-viewpoint image synthesis using superpixel segmentation,'' *APSIPA Trans. Signal Inf. Process.*, vol. 6, 2017. [Online]. Available: https://www.cambridge.org/core/journals/apsipa-transactions-on-signal-and-information-processing/volume/9D4FF8C693586983FE5D764C8DCEE30C, doi: 10.1017/ATSIP.2017.5.
[49] A. Telea, ''An image inpainting technique based on the fast marching method,'' *J. Graph. Tools*, vol. 9, no. 1, pp. 23–34, Jan. 2004, doi: 10.1080/10867651.2004.10487596.
[50] A. Newson, A. Almansa, Y. Gousseau, and P. Pérez, ''Non-local patch-based image inpainting,'' *Image Process. On Line*, vol. 7, pp. 373–385, Dec. 2017, doi: 10.5201/ipol.2017.189.
[51] D. Pathak, P. Krähenbuhl, J. Donahue, T. Darrell, and A. A. Efros, ''Context encoders: Feature learning by inpainting,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2536–2544. [Online]. Available: https://ieeexplore.ieee.org/document/7780647/
[52] Y. Wexler, E. Shechtman, and M. Irani, ''Space-time completion of video,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 3, pp. 463–476, Mar. 2007. [Online]. Available: https://ieeexplore.ieee.org/document/4069262/
[53] S. Li, C. Zhu, and M.-T. Sun, ''Hole filling with multiple reference views in DIBR view synthesis,'' *IEEE Trans. Multimedia*, vol. 20, no. 8, pp. 1948–1959, Aug. 2018, doi: 10.1109/TMM.2018.2791810.
[54] S. Fachada, D. Bonatto, A. Schenkel, and G. Lafruit, ''Depth image based view synthesis with multiple reference views for virtual reality,'' in *Proc. IEEE 3DTV-Conf., True Vis.-Capture, Transmiss. Display 3D Video (3DTV-CON)*, Jun. 2018, pp. 1–4, doi: 10.1109/3DTV.2018.8478484.
[55] R. Pintus, E. Gobbetti, and M. Callieri, ''Fast low-memory seamless photo blending on massive point clouds using a streaming framework,'' *J. Comput. Cultural Heritage*, vol. 4, no. 2, pp. 1–15, Nov. 2011. [Online]. Available: https://dl.acm.org/citation.cfm?doid=2037820.2037823
[56] A. Dziembowski and M. Domański, ''Adaptive color correction in virtual view synthesis,'' in *Proc. IEEE 3DTV Conf., True Vis.-Capture, Transmiss. Display 3D Video (3DTV-CON)*, Jun. 2018, pp. 1–4, doi: 10.1109/3DTV.2018.8478439.
[57] S.-P. Lu, B. Ceulemans, A. Munteanu, and P. Schelkens, ''Spatio-temporally consistent color and structure optimization for multiview video color correction,'' *IEEE Trans. Multimedia*, vol. 17, no. 5, pp. 577–590, May 2015, doi: 10.1109/TMM.2015.2412879.
[58] S. Ye, S.-P. Lu, and A. Munteanu, ''Color correction for large-baseline multiview video,'' *Signal Process., Image Commun.*, vol. 53, pp. 40–50, Apr. 2017, doi: 10.1016/j.image.2017.01.004.
[59] B. Salahieh, S. Bathia, J. Boyce, J. Fleureau, and R. Doré, *Hybrid Multi-Pass Add-on Tool for Coherent and Complete View Synthesis of Natural Content [M47502]*, document ISO/IEC JTC1/SC29/WG11, Mar. 2019, p. 11.
[60] Z. Zhang, ''Microsoft Kinect sensor and its effect,'' *IEEE MultiMedia*, vol. 19, no. 2, pp. 4–10, Feb. 2012.
[61] T. Senoh, K. Yamamoto, N. Tetsutani, and H. Yasuda, *Enhanced DERS for Quad Reference Views (eDERS)*, document ISO/IEC JTC1/SC29/WG11, Jan. 2018.
[62] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, ''A naturalistic open source movie for optical flow evaluation,'' in *Proc. 12th Eur. Conf. Comput. Vis. (ECCV)*, Oct. 2012, pp. 611–625, doi: 10.1007/978-3-642-33783-3_44.
[63] J. Jung, B. Kroon, R. Doré, G. Lafruit, and J. Boyce, *CTC on 3DoF+ and Windowed 6DOF (v2) [N17726]*, document ISO/IEC JTC1/SC29/WG11, Jul. 2018.
[64] B. Salahieh, B. Kroon, J. Jung, and M. Domanski, *Test Model 4 for Immersive Video [N19002]*, document ISO/IEC JTC1/SC29/WG11, Feb. 2020.
[65] D. Doyen, T. Langlois, B. Vandame, F. Babon, G. Boisson, N. Sabater, R. Gendrot, and A. Schubert, *Light Field Content From 16-Camera Rig [M40010]*, document ISO/IEC JTC1/SC29/WG11, Jan. 2017.
[66] M. Domanski, A. Dziembowski, A. Grzelka, D. Mieloch, O. Stankiewicz, and K. Wegner, *Multiview Test Video Sequences for Free Navigation Exploration Obtained Using Pairs of Cameras [M38247]*, document ISO/IEC JTC1/SC29/WG11, May 2016.
[67] K. Wegner, *Software Manual DERS*, document ISO/IEC JTC1/SC29/WG11, 2014, p. 17.
[68] T. Tezuka, M. P. Tehrani, K. Suzuki, K. Takahashi, and T. Fujii, ''View synthesis using superpixel based inpainting capable of occlusion handling and hole filling,'' in *Proc. Picture Coding Symp. (PCS)*, May 2015, pp. 124–128, doi: 10.1109/PCS.2015.7170060.
[69] D. Bonatto, S. Fachada, and G. Lafruit, ''RaViS: Real-time accelerated view synthesizer for immersive video 6DoF VR,'' in *Proc. IS&T Int. Symp. Electron. Imag., Sci. Technol.*, Society for Imaging Science and Technology, Jan. 2020, doi: 10.2352/ISSN.2470-1173.2020.13.ERVR-382.
[70] G. Riegler and V. Koltun, ''Free view synthesis,'' in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2020, pp. 623–640, doi: 10.1007/978-3-030-58529-7_37.
[71] S. Kwak, J. Yun, J. Y. Jeong, W.-S. Cheong, and J. Seo, *Ray-Based Blending Weight for 6DoF View Synthesis [M54409]*, document ISO/IEC JTC1/SC29/WG11, Jun. 2020, p. 5.
[72] J. L. Schönberger, E. Zheng, J.-M. Frahm, and M. Pollefeys, ''Pixelwise view selection for unstructured multi-view stereo,'' in *Computer Vision* (Lecture Notes in Computer Science), vol. 9907, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham, Switzerland: Springer, 2016, pp. 501–518, doi: 10.1007/978-3-319-46487-9_31.
[73] D. S. K. Fok and D. Crevier, ''Volume estimation by Monte Carlo methods,'' *J. Stat. Comput. Simul.*, vol. 31, no. 4, pp. 223–235, Apr. 1989, doi: 10.1080/00949658908811145.
[74] M. Regan and G. S. P. Miller, ''The problem of persistence with rotating displays,'' *IEEE Trans. Vis. Comput. Graphics*, vol. 23, no. 4, pp. 1295–1301, Apr. 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7829409/

[75] G. Denes, K. Maruszczyk, G. Ash, and R. K. Mantiuk, "Temporal resolution multiplexing: Exploiting the limitations of spatio-temporal vision for more efficient VR rendering," *IEEE Trans. Vis. Comput. Graphics*, vol. 25, no. 5, pp. 2072–2082, May 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8643566/

[76] Y. Kuroki, T. Nishi, S. Kobayashi, H. Oyaizu, and S. Yoshimura, "Improvement of motion image quality by high frame rate," in *SID Symp. Dig. Tech. Papers*, vol. 37, no. 1, 2006, p. 14, doi: 10.1889/1.2433276.

[77] V. Corporation. (Nov. 2018). *Valvesoftware/OpenVR*. [Online]. Available: https://github.com/ValveSoftware/openvr

[78] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *Proc. 37th Asilomar Conf. Signals, Syst. Comput.*, 2003, pp. 1398–1402. [Online]. Available: https://ieeexplore.ieee.org/document/1292216/

[79] S. Rogge, I. Schiopu, and A. Munteanu, "Depth estimation for light-field images using stereo matching and convolutional neural networks," *Sensors*, vol. 20, no. 21, p. 6188, Oct. 2020, doi: 10.3390/s20216188.

[80] S. Rogge and A. Munteanu, "Depth estimation in light field camera arrays based on multi-stereo matching and belief propagation," in *Proc. 3DTV-Conf., True Vis. Capture, Transmiss. Display 3D Video (3DTV-CON)*, Jun. 2018, pp. 1–4, doi: 10.1109/3DTV.2018.8478503.

[81] S. Rogge, B. Ceulemans, Q. Bolsee, and A. Munteanu, "Multi-stereo matching for light field camera arrays," in *Proc. 26th Eur. Signal Process. Conf. (EUSIPCO)*, Sep. 2018, pp. 251–255, doi: 10.23919/EUSIPCO.2018.8553075.

[82] A. Dziembowski, *Software Manual of IV-PSNR for Immersive Video [N19495]*, document ISO/IEC JTC1/SC29/WG11, Jul. 2020.

[83] J. Jung, G. Lafruit, and A. Schenkel, *Exploration Experiments for MPEG-I: Windowed-6DoF [N17609]*, document ISO/IEC JTC1/SC29/WG11, Apr. 2018.

[84] A. Schenkel, S. Fachada, D. Bonatto, and G. Lafruit, *Response to Exploration Experiments for MPEG-I Windowed-6DoF [M43509]*, document ISO/IEC JTC1/SC29/WG11, Jul. 2018.

[85] J. T. Kajiya and B. P. Von Herzen, "Ray tracing volume densities," *ACM SIGGRAPH Comput. Graph.*, vol. 18, no. 3, pp. 165–174, Jul. 1984, doi: 10.1145/964965.808594.

[86] P. Ndjiki-Nya, M. Koppel, D. Doshkov, H. Lakshman, P. Merkle, K. Muller, and T. Wiegand, "Depth image-based rendering with advanced texture synthesis for 3-D video," *IEEE Trans. Multimedia*, vol. 13, no. 3, pp. 453–465, Jun. 2011. [Online]. Available: https://doi.org/10.1109/TMM.2011.2128862

[87] S. Fachada and G. Lafruit, "Computer generated holography with depth-based view synthesis," in *Proc. OSA Imag. and Appl. Opt. Congr.*, Jun. 2020, p. 3, doi: 10.1364/DH.2020.HF1D.7.

[88] S. Fachada, D. Bonatto, and G. Lafruit, "High-quality holographic stereogram generation using four RGBD images," *Appl. Opt.*, vol. 60, pp. A250–A259, Oct. 2020. [Online]. Available: https://www.osapublishing.org/ao/abstract.cfm?doi=10.1364/AO.403787

[89] S. Lu, J. Hanca, A. Munteanu, and P. Schelkens, "Depth-based view synthesis using pixel-level image inpainting," in *Proc. 18th Int. Conf. Digit. Signal Process. (DSP)*, Jul. 2013, pp. 1–6, doi: 10.1109/ICSDP.2013.6622773.

[90] G. Wadhwa, A. Dhall, S. Murala, and U. Tariq, "Hyperrealistic image inpainting with hypergraphs," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2021, pp. 3912–3921.

DANIELE BONATTO received the master's degree in computational intelligence software engineering in applied sciences from the Université Libre de Bruxelles, Brussels, Belgium, in 2016. He is currently pursuing the Ph.D. degree in real-time 3D computing, jointly between the Université Libre de Bruxelles and the Vrije Universiteit Brussel, respectively, the French and Dutch wing of the Free University of Brussels, Brussels. He works on real-time free-viewpoint rendering of natural scenery with sparse multi-camera acquisition setups. Jointly with the Moving Picture Experts Group (MPEG), he developed the reference view synthesis software, in 2018, and two high-density natural scene data sets for static and dynamic content explorations.

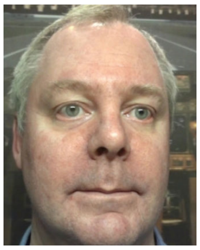
SARAH FACHADA (Student Member, IEEE) graduated from the Ecole polytechnique, France, and the Trinity College of Dublin, Ireland in 2017, majoring in computer science. She is currently pursuing the Ph.D. degree with the Université Libre de Bruxelles, Belgium, working on acquisition and rendering in light fields and DIBR. She is a Research Fellow of the Fonds de la Recherche Scientifique—FNRS, Belgium. She explores fields, such as rendering with non-pinhole cameras, geometric algebra applications and rendering non-Lambertian objects. Jointly with MPEG, she developed the reference view synthesis software, in 2018, and dynamic natural scene datasets.

SÉGOLÈNE ROGGE received the degree in computational intelligence software engineering in applied sciences from the Université Libre de Bruxelles, Brussels, Belgium, in 2016. She is currently pursuing the Ph.D. degree entitled *3D reconstruction using multimodal camera systems* at the Vrije Universiteit Brussel and the Université Libre de Bruxelles, Belgium. Jointly with the Moving Picture Experts Group (MPEG), she is involved with the reference depth estimation software and her actual focus is on depth estimation for light field cameras.

ADRIAN MUNTEANU (Member, IEEE) received the M.Sc. degree in electronics and telecommunications from the Politehnica University of Bucharest, Romania, in 1994, the M.Sc. degree in biomedical engineering from the University of Patras, Greece, in 1996, and the Ph.D. degree in applied sciences from Vrije Universiteit Brussel (VUB), Belgium, in 2003.

From 2004 to 2010, he was a Postdoctoral Fellow with the Fund for Scientific Research–Flanders (FWO), Belgium, and since 2007, he has been a Professor at VUB. He is currently a Professor with the Electronics and Informatics (ETRO) Department, VUB. His research interests include image, video and 3D graphics compression, 3D video, deep learning, distributed visual processing, error-resilient coding, and multimedia transmission over networks. He is the author of more than 350 journal articles and conference publications, book chapters, and contributions to standards, and holds seven patents in image and video coding.

Dr. Munteanu was a recipient of the 2004 BARCO-FWO prize for his Ph.D. work, a (co-)recipient of the Most Cited Paper Award from Elsevier for 2007, and of ten other scientific prizes and awards. He served as an Associate Editor for the IEEE Transactions on Multimedia. He also serves as an Associate Editor for the IEEE Transactions on Image Processing.



GAUTHIER LAFRUIT (Member, IEEE) received the M.Sc. and Ph.D. degrees from Vrije Universiteit Brussel (VUB), in 1989 and 1995, respectively. He is currently an Associate Professor of virtual reality and light field technologies at the l'Université Libre de Bruxelles (ULB), Brussels, Belgium. He has worked for 25 years in the domain of visual data analysis and compression, participating to compression standardization committees like CCSDS (space applications), JPEG (still picture coding) and MPEG (moving picture coding). His research interests include depth image-based rendering, immersive video, point-clouds, and digital holography technologies.