Motion parallax for 360° RGBD video

Ana Serrano, Incheol Kim, Zhili Chen, Stephen DiVerdi, Diego Gutierrez, Aaron Hertzmann, and Belen Masia

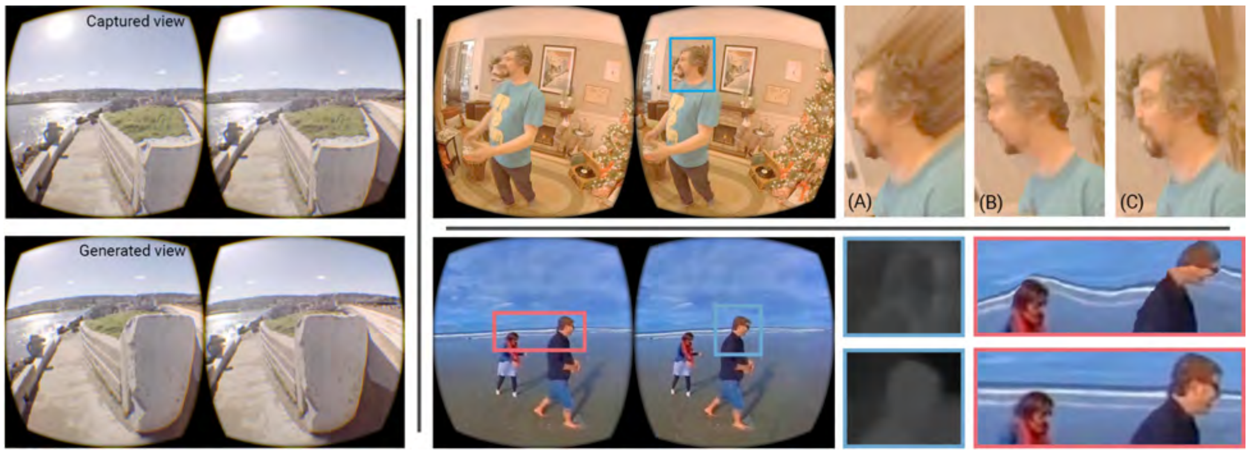

Fig. 1. We add parallax for 360° videos, for viewing in virtual reality head-mounted displays (HMDs). This translates into a more compelling viewing experience, as our user studies confirm. *Left*: Captured point of view as shown in the HMD (top), and a novel view as the user moves their head (bottom); this novel view is generated with our method and was not captured by the camera. *Right, top row*: Straightforward approaches based on image-based rendering do not work well due to suboptimal quality of the depth information. Original view (left) captured with the GoPro Odyssey, and a close-up of novel views generated with three different methods (right): (A) naive reprojection of RGB information, (B) naive handling of disocclusions, and (C) our method, relying on a robust layered representation. *Right, bottom row*: We also propose a depth map improvement step to correct for errors that have a high impact on reprojection. Original view (left) from a YouTube video (`https://youtu.be/iWyvlkWYXhY`), and close-ups showing depth maps and a displaced view computed with them, for the original estimated depth map (top row), and for our improved depth map (bottom row).

Abstract—We present a method for adding parallax and real-time playback of 360° videos in Virtual Reality headsets. In current video players, the playback does not respond to translational head movement, which reduces the feeling of immersion, and causes motion sickness for some viewers. Given a 360° video and its corresponding depth (provided by current stereo 360° stitching algorithms), a naive image-based rendering approach would use the depth to generate a 3D mesh around the viewer, then translate it appropriately as the viewer moves their head. However, this approach breaks at depth discontinuities, showing visible distortions, whereas cutting the mesh at such discontinuities leads to ragged silhouettes and holes at disocclusions. We address these issues by improving the given initial depth map to yield cleaner, more natural silhouettes. We rely on a three-layer scene representation, made up of a foreground layer and two static background layers, to handle disocclusions by propagating information from multiple frames for the first background layer, and then inpainting for the second one. Our system works with input from many of today's most popular 360° stereo capture devices (e.g., Yi Halo or GoPro Odyssey), and works well even if the original video does not provide depth information. Our user studies confirm that our method provides a more compelling viewing experience than without parallax, increasing immersion while reducing discomfort and nausea.

Index Terms—Immersive environments, Virtual Reality video.

1 Introduction

With the growth of Virtual Reality (VR) headsets, stereo 360 video is becoming increasingly popular. Existing devices, including the GoPro Odyssey, Yi Halo, Vuze VR, Jaunt ONE or Facebook Surround 360, capture video that is converted to a stereo 360° video format for viewing in headsets. This video creates a feeling of "immersion" because it fills the viewer's field of view. However, this representation does not support motion parallax when the viewer shifts their head translationally. This experience is unnatural, can break the sense of immersion, and can cause discomfort and even nausea in some users, due to the mismatch between the visual and vestibular systems [67]. Even if the viewer attempts to stay static and carry out only rotational movements, accidental motion is likely to happen, leading to potential discomfort and sickness. Viewers that can display imagery with rotation and translation are commonly referred to as *6-DoF*; correspondingly, rotation-only viewing is referred to as *3-DoF*.

HMD-ready demos exhibiting motion parallax in synthetic and real world scenarios, such as *Welcome to Light Fields* [52], have shown the potential of 6-DoF viewing. Experiments have been carried out comparing 3-DoF and 6-DoF viewing, showing the importance of motion parallax to the viewing experience in VR, and how 6-DoF viewing leads to higher immersion and realism, and lower discomfort [57,67]. At present, however, there is no practical system for capturing general-purpose 6-DoF video. While a number of research prototypes have been demonstrated, most of them only work for static scenes (e.g., [32,48]), require impractical amounts of video storage for a reasonable range of head motion [58], require complex indoor setups and only capture actors within a constrained volume [71], or, else are still only

• *Ana Serrano, Incheol Kim, Diego Gutierrez, and Belen Masia are with Universidad de Zaragoza, I3A.*
• *Zhili Chen, Stephen DiVerdi, and Aaron Hertzmann are with Adobe Research.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

proofs-of-concept that—as well as can be determined from available information—seem to require large, expensive setups [36].

We introduce a new approach to adding motion parallax to footage recorded by existing 360° video capture systems. We first obtain a suitable depth map from the input video. An initial depth can be provided by existing 360 stitching algorithms (e.g., [1]), or we can estimate it using an off-the-shelf deep learning algorithm [27]. A baseline approach would be to directly create a 3D scene from this initial depth map. However, because conventional depth map algorithms are not designed for reprojection, the baseline approach creates objectionable artifacts (Figure 1), including lack of disocclusions, jagged boundaries, and lack of temporal coherence.

Our approach is as follows. We introduce a three-layer representation of the video for playback, which allows disocclusions for both moving and static scene elements (Section 3.1). The input video plus depth is used as the foreground layer. Object silhouettes are detected and cut in this layer, to allow for disocclusions. One background layer is computed by inpainting behind static occluders, and the other by background subtraction behind moving occluders. Since the depth maps provided by existing algorithms are not suitable for reprojection [68], we also introduce an algorithm that preprocesses such depth maps to minimize visible artifacts (Section 4). The algorithm cleans up occlusion boundaries and improves temporal coherence, leading to more visually plausible and appealing results.

Our approach provides a drop-in 6-DoF viewing experience that works for a wide class of existing 360° video available today, without any new capture or hardware requirements; we only assume that the video is captured by a static camera, which is common practice to minimize sickness during viewing. Although our method is not completely free of artifacts, we have performed three different user studies (Section 5), which confirm that it creates a more natural, compelling viewing experience, while also reducing the feeling of sickness or discomfort compared to traditional 3-DoF viewing. We provide source code for both our preprocessing algorithm and our real-time layered video viewer[1].

2 Related Work

Image-based rendering. Since the seminal works on image-based rendering [8, 19, 29, 50, 59], a number of IBR techniques have emerged that differ mainly either in the characteristics of the input data, or the type of scene representation used. Our work is related to this field, but our input differs substantially from what these works typically use.

A large group of works seek to allow free viewpoint navigation performing reprojection aided by some geometry proxy, often employing multiview stereo to obtain a 3D reconstruction [13, 19, 23]. To compensate for potential errors or sparsity in the 3D reconstruction, Goesele et al. [28] use *ambient point clouds* when rendering areas of the scene that are poorly reconstructed, while Chaurasia et al. [14] use variational image warping to compensate for sparse 3D information. Baricevic et al. [3] densify sparse depth information from monocular SLAM via an optimization approach in a live scene, i.e., without pre-capture or preprocessing. All these works are targeted at multiview setups with a relatively large baseline, while our input is an RGBD video panorama. Still, we use some ideas from these works, like the need for soft visibility maps at depth boundaries to reduce artifacts [23, 53].

Some works do not rely on an explicit 3D reconstruction, but rather on dense image correspondences, for view interpolation [46, 49]. In contrast to them, we have the ability of generating novel unseen viewpoints, rather than interpolating between existing ones. Others augment well-known RGB inpainting methods [16] with depth-based terms [9, 10] for disocclusion filling in novel views; we partially rely on inpainting, but will show that simpler methods are better suited for our purposes.

Learning-based approaches have also been used to interpolate novel views, e.g. [26], or for light field acquisition [38], including light field video [69]. Closer to ours is the recent work by Zhou and colleagues [70] for wide-baseline stereo pair generation from narrow-baseline stereo input, using a deep neural network that infers a multi-plane image representation of the scene; however, they do not handle dynamic scenes and their method is limited to stereo pairs.

The use of a layered representation is common in IBR approaches. Zitnick et al. [71] use a two-layer representation, in a system with a fixed, wide-baseline camera rig that allows them to obtain cleaner depth boundaries and mattes for areas near depth discontinuities while handling video. A more recent example is the work of Hedman et al. [32], who also use a two-layer representation for rendering. Follow-up work [33] improves reconstruction quality and computation times leveraging dual-lens cameras present in current high-end mobile devices. These two methods capture high-fidelity *static* 3D scenes, but they are not suitable for dynamic scenes nor video since the scene needs to be captured from many different points of view (some of the examples they provide in the dataset require dozens of images as input). In contrast, our approach allows automatic 6-DoF capture for dynamic scenes and videos, being agnostic to the capture hardware, including capture setups with a very narrow baseline. Another interesting approach is that of Philip and Drettakis [54], in which multi-view inpainting is performed on an intermediate representation formed by locally planar spaces shared between the input images; the work focuses on inpainting large unknown regions in large datasets with wider baselines than ours.

Finally, a series of works perform novel view synthesis to create content for autostereoscopic displays from stereo pairs, via image-domain warping [63]; using a steerable pyramid decomposition and filtering also used in motion magnification [21]; dealing with artifact detection and removal in synthesized stereo pairs [20]; or focusing on real time performance during the conversion [39]. In our case, the novel views we create can be from any viewpoint and do not follow a certain structure as they usually do in autostereoscopic displays.

6-DoF viewing. A number of works have targeted creating a full 6-DoF experience in VR. Thatte et al. [66] propose a novel data representation to support motion parallax, *depth augmented stereo panoramas*, but it requires a specific capture setup, different from that of commonly available footage. Huang et al. [35] take as input a monoscopic 360° video with a moving camera, and, after doing a 3D reconstruction of the scene, are able to generate novel viewpoints by warping the initial 360° footage. They require, however, that the movement of the camera in the input video provides sufficient baseline for the 3D reconstruction. Visualization of real 360° scenes with head motion parallax is provided by the work of Luo et al. [48]; however, they require that a robotic camera arm captures the scene at specific positions, uniformly sampling a sphere in latitude and longitude, and they cannot handle dynamic scenes. Also addressing the problem of 6-DoF 360° content from light fields, Hinds et al. [34] proposed a benchmark for assessing light field compression techniques in the context of a view synthesis pipeline.

There has also been work on applying view-dependent texture mapping [18, 19] to speed up the rendering of the stereo pair in VR setups, assuming the input is a set of captured images suitable for a multiview stereo technique [56]. The recent work of Koniaris et al. [41] provides head motion parallax and view-dependent lighting effects in 360° content, but is targeted at synthetic content: the method takes as input a number of pre-rendered views of the scene of interest, as opposed to our real-world captured videos. Finally, Schroers et al. [58] presented a system that enables horizontal motion parallax from footage captured with a 16-camera rig: They reconstruct a dense light field panorama from a sparse number of input views. There are, however, three main differences with respect to ours: First, they require a calibrated camera rig for capture; second, they require a much larger amount of storage, since they need to store a separate 360° video stream for *every* discrete viewpoint; and third, they only provide horizontal parallax, which has been shown to yield a significantly worse viewing experience than full 6-DoF parallax [67].

In contrast to these works, we take as input a monoscopic 360° video of a scene captured from a static viewpoint, as given by typical 360° capture systems, plus a depth map obtained from the recorded footage. To our knowledge, only the work of Sayyad et al. [57] targets providing 6-DoF from a single panorama, but they do it via user intervention through an interactive modeling tool that allows the user to modify the

[1]http://webdiis.unizar.es/∼aserrano/projects/VR-6dof

geometry directly in VR; in contrast, we target an automatic approach that can also be suitable for video and dynamic scenes.

Depth map improvement techniques. Due to small baselines, specular or transmissive surfaces, or moving objects, among other causes, depth maps estimated from multiple images can be imprecise and contain artifacts. Image-guided filtering techniques provide depth maps that are well-aligned with the corresponding RGB edges; examples of local methods that can be used include bilateral filtering [4, 15], joint bilateral upsampling [42], guided filtering [31], or multilateral filtering [12]. However, RGB discontinuities may lead to depth artifacts that become clearly visible in 6-DoF viewing.

Global optimization methods can also be used for depth refinement. Levin et al.'s colorization based on optimization [44], which propagates scribbles according to the color affinity in the input image, has been applied to depth maps [32]. Image matting can also be used to propagate input scribbles [45]. These methods obtain good results for hole filling and sharp input edges, but this is not always the case in estimated depth maps from camera rigs or single-image methods.

We propose here a depth refinement technique that is inspired by these works but tailored for our end goal. For a more in-depth review of related techniques, we refer the reader to a recent survey on hole filling strategies for depth map completion [2].

There is another line of works that try to minimize contour artifacts when *reconstructing* depth from stereo or multiview content [6, 25, 61]. However, our input already includes a depth estimate along with the RGB frames, while building it from scratch applying these existing methods would likely fail due to the small baseline and minimal overlapping regions of current 360° cameras; for example, the work of Shan et al. [61] requires thousands of images. Further, this would limit our generality, since it would not work with monocular footage. Additionally, the works of Feris et al. [25], and Birchfield and Tomasi [6] are not intended for reprojection, and thus they do not consider geometry appearance from novel points of view.

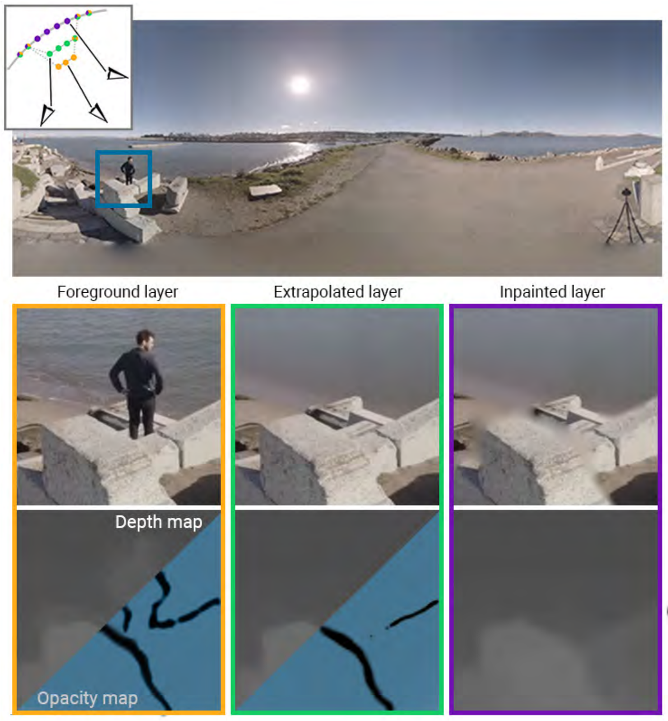

Fig. 2. Example showing the layers in our scene representation. Top row: Sample RGB frame. The inset depicts a simple illustration of our three layers, showing which layer the user will see from different points of view (for a more detailed explanation see Figure 3, left). Middle row: For a close-up region, we show RGB of each layer (*foreground*, *extrapolated*, and *inpainted*). Bottom row: For the same close-up region, we show the associated depth map, and, for the two layers where it is present, the associated opacity map.

3 Layered video representation

The input to our method is a 360° video, with RGB and depth values at each pixel. These can be initially provided by existing stitching algorithms [1], or estimated with a CNN [27]. Instead of using this depth information as is, which leads to visible artifacts when enabling 6-DoF, we first preprocess it and make it suitable for reprojection (see Section 4).

The most basic reprojection algorithm is to convert the input video into a spherical triangle mesh, mapping each pixel to a vertex with the given RGB value, and 3D position as determined by the depth map, i.e., given a pixel at coordinates (θ, ϕ) on the equirectangular image, and depth d, map it to spherical coordinates (d, θ, ϕ). We use input videos with resolution 1024×2048, so this corresponds to a mesh of 1024×2048 vertices. Then, this mesh can be rendered at runtime, as the viewer moves their head. However, this *naive reprojection* approach produces very noticeable artifacts at disocclusion boundaries, as shown in Figure 1, close-up (A): As the viewer moves away from the center of the projection, the triangles of the mesh that correspond to disocclusion boundaries incorrectly connect foreground and background elements, rather than revealing disoccluded regions. To fix this, one might attempt to identify disocclusion boundaries, say, by depth differences, and then break the mesh at those boundaries. However, this naive handling of disocclusions can lead to jagged silhouette boundaries or missing information due to inaccuracies in the boundary estimation, in the depth map, or in the correspondence between depth and RGB edges, as shown in Figure 1, close-up (B).

We do employ a mesh-based approach, but making use of a layered representation that allows us to fix the aforementioned issues of missing information and jagged silhouettes in disocclusion boundaries. We describe this representation first (Section 3.1), then how it is computed (Section 3.2), and finally, how this representation is used to render the scene during real-time viewing (Section 3.3).

3.1 Scene representation

Our scene representation is designed to display the RGBD video with clean disoclussion boundaries at object silhouettes. In order to fill holes (missing information) in disocclusions, we use both background subtraction and inpainting where appropriate. We seek, as much as possible, to use the original video for rendering, e.g., rather than splitting objects in the video into separate layers. Finally, we wish to allow for real-time playback.

In order to achieve these goals, we introduce a layered representation. Our representation extends previous layered IBR approaches (e.g., [32, 71]); specifically, we extend the representation to three layers: a dynamic foreground layer, and two static background layers. The *foreground* layer is a mesh generated from the RGB video and its associated depth, together with an extra per-frame opacity map ($\hat{\alpha}^F$) used to control the opacity of the mesh at disocclusion boundaries. This layer is thus stored as an RGB video with an associated depth video and opacity video. There are two static background layers (which we term the *extrapolated layer* and the *inpainted layer*, respectively), used to fill disocclusion holes created when motion parallax takes place as the viewer moves their head. The extrapolated layer contains information for static background regions that are, at some time, occluded by moving objects (e.g., behind a moving person or car); hence, it can be computed by background subtraction. It also has a static opacity map ($\hat{\alpha}^E$) associated with it, again used to control transparency of this mesh when disocclusions occur. The layer is stored as an RGBA image and associated depth map. The inpainted layer contains information to fill-in disocclusions of areas corresponding to static background regions occluded by static objects. Those areas were thus never observed by the camera, and so must be computed by inpainting. Since this is the rearmost layer, it does not have an associated opacity map, and is stored as an RGB image and associated depth map. An example of these three

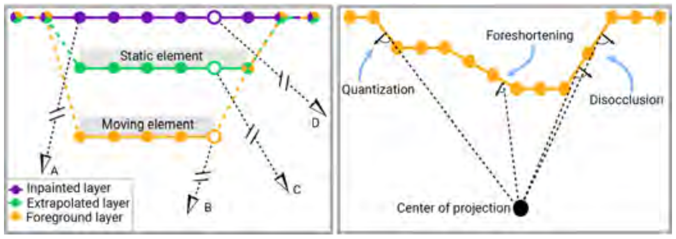

Fig. 3. Left: Illustration explaining our three-layer representation. Each circle represents a different pixel, and its color indicates the layer it belongs to. Lines represent mesh connectivity (i.e., faces), and potential disocclusions are represented with dashed lines. The *foreground* layer shows a moving object and the *extrapolated* layer a static one. Some simple cases can be handled with only two layers (viewpoint A; black dotted lines represent a single ray for a given viewpoint). In others, however, three layers are required because depending on the viewpoint we the user should see the *foreground layer* (viewpoint B), the *extrapolated layer* (viewpoint C), or the *inpainted layer* (viewpoint D). Right: Illustration showing the relationship between face normals and potential disocclusions. The angle between these normals and the viewing direction from the center of projection (dashed lines indicate this direction for a number of faces) is used to compute the initial opacity map of a layer (O^F). We identify potential disocclusions as mesh triangles with angles close to $90°$. Angles due to foreshortening will be narrower than angles belonging to disocclusions, since depth discontinuities are smoother. Quantization errors will produce sparse angles close to $90°$.

layers is shown in Figure 2, and a visualization is given in Figure 3.

Given this representation, rendering the scene at a given time index entails converting the three RGBD images to three meshes as described above, and rendering each in the same 3D space. While the viewer's head remains at the center of projection, only the foreground layer will be seen. When the viewer's head moves from the camera center, the background layers will be visible at disocclusions. To enable this, the transparency of the foreground layer and extrapolated layer is computed making use of the opacity maps ($\hat{\alpha}^F$ and $\hat{\alpha}^E$, respectively). Section 3.3 describes rendering of the meshes during real-time playback.

Note that the static background layers cannot be merged because both layers may contain information in a given (θ,ϕ) direction. For example, suppose a person walks in front of a static car. From the camera center, three layers overlap: the moving person, which will be rendered in the *foreground layer*; the static car, which must be rendered in the *extrapolated layer*, since it is visible in other video frames; the scene behind the car, estimated by inpainting into the *inpainted layer*. This is illustrated in Figure 3 (left): depending on the user's point of view as they move, we will need to show either the *foreground layer* (viewpoint B), the *extrapolated layer* (viewpoint C), or the *inpainted layer* (viewpoint D).

3.2 Layer computation

We now describe how the layered representation is computed, including the three layers and the opacity maps.

3.2.1 Foreground layer

The foremost layer is dynamic and is the only one visible as long as the viewer's head does not move from the center of projection. It is directly generated from the original RGB video, together with the pre-processed depth (the pre-processing of the original depth is described in Section 4).

The opacity map stores, for each vertex of the *foreground layer* mesh, a value $\hat{\alpha}^F \in [0..1]$ that will be used to control the opacity of the layer at runtime (Section 3.3). The idea is that, as the viewer's head moves from the center of projection, and disocclusions occur in certain areas, the *foreground layer* fades in those areas to allow visibility of the back layers. Thus, this map should store which vertices are likely to belong to disocclusion boundaries. Intuitively, disocclusion boundaries will be found at parts of the mesh whose normals are approximately perpendicular to the *viewing direction from the center of projection*, since that is indicative of a sharp depth discontinuity and thus a potential disocclusion boundary (see Figure 3, right). Thus, by storing the angles between face normals and the view direction from the center of projection (in practice, the dot product between these two vectors) we would in principle have an initial foreground layer opacity map per frame, O^F. This initial opacity map, which is different for every frame of the input video, is shown in Figure 4(b) for a sample frame. Note that we leverage the GPU rendering pipeline, and compute the dot product in the fragment shader, i.e., with fragment normals (given the topology of our meshes, for each layer there is a single fragment per pixel of the input frame).

However, as also shown in Figure 3 (right), angles close to perpendicular can also be the consequence of foreshortening, or the result of quantization errors in the depth map. To minimize the impact of these situations, we apply to this initial opacity map O^F a closing morphological operation to remove subtle orientation changes due to depth artifacts, followed by a thresholding operation to remove smooth depth variations (most likely due to foreshortening or quantization effects), and then blur the result to provide a smooth transition at boundaries. Finally, a logistic function is applied in order to remove intermediate values that can cause ghosting, while still allowing for a smooth transition. Thus, the final opacity map of the *foreground layer*, $\hat{\alpha}^F$, is computed as:

$$\hat{\alpha}^F = S(G \circledast (\tau(O^F \bullet K))) \tag{1}$$

Operator $\bullet$ denotes the closing morphological operation, K is a disk kernel (radius of 2), τ denotes the thresholding operation (we use 0.8 as a threshold in all cases since O^F is in the range $[0..1]$, 1 meaning perpendicular orientation), G is a Gaussian blur operator (kernel size of 7×7), and S is a logistic function (centered at $c = 0.5$, and with a slope $k = 8$), shown in Eq. 2. Note that we apply all the operations in Eq. 1 on the complement of O^F, and then take the complement of the result to yield $\hat{\alpha}^F$.

$$S(x) = \frac{1}{1+e^{-k(x-c)}} \tag{2}$$

While we define our parameters heuristically, we use the same parameters for all input videos and all the different stitching methods and cameras we tested. Figures 4(c) and 4(d) show the result of applying these operations, while in the supplementary material we include individual results for each of the operations described.

3.2.2 Extrapolated layer

This layer is static, and essentially contains a version of the scene with moving objects removed. We first compute its depth map, by taking, for each pixel, the n largest depth values of the *foreground* video frames, and compute their median to obtain a robust maximum depth (in all our tests, $n = 15$). Setting the depth to this robust maximum ensures that the observed background layer will remain behind the foreground layer, and thus invisible from the original camera viewpoint. To obtain the corresponding RGB values, we take, for each pixel, the RGB value associated to the selected depth value.

The computation of the opacity map for the *extrapolated layer*, $\hat{\alpha}^E$, is analogous to that of the *foreground layer opacity map* previously described. The only difference is that in this case we use the mesh of the *extrapolated layer* when computing the angles between the mesh normals and the view direction from the center of projection to yield the initial opacity map O^E. As such, this opacity map does not need to be computed per frame of the input video, but just once.

3.2.3 Inpainted layer

This is the backmost layer; it is also static, and contains inpainted regions in areas behind static objects that can become visible due to disocclusions. We obtain this layer by identifying which regions can become disoccluded, and inpainting them. The identification is done based on the opacity maps, whose computation is described below. A theoretical derivation to compute the maximum parallax that the layers may be subject to, and thus the largest area requiring inpainting, can be found in the supplementary material. For inpainting we use a

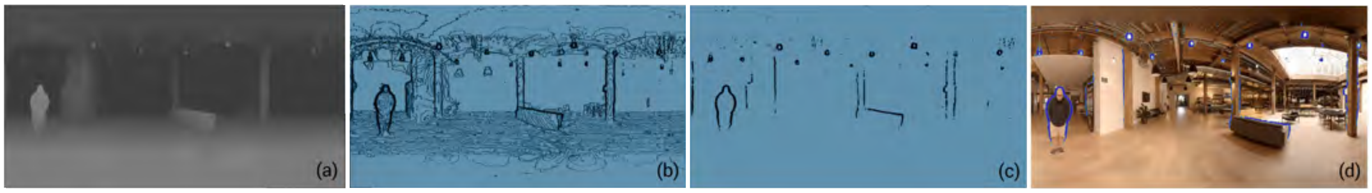

Fig. 4. Example showing the computation of the opacity map. (a) Depth map of the *foreground layer* for a given frame of the video. (b) Initial opacity map O^F. (c) Final opacity map $\hat{\alpha}^F$. (d) Final opacity map shown in (c), overlaid on top of the corresponding RGB image. The orientation values O^F shown in (b) are too noisy to provide smooth opacity values, as they include information not only from potential disocclusions, but also from areas with foreshortening and quantization errors. We process these raw values (see text for details), and obtain a clean opacity map $\hat{\alpha}^F$, shown in (c), that smoothly matches potential disocclusion boundaries in the RGB image, shown in (d).

well-established PDE-based inpainting method [5] that smoothly interpolates the unknown values. We favor smoothing approaches over patch-based ones, since, upon failure, the latter are prone to produce very prominent local artifacts that are very distracting during playback, making our smooth approximation preferable for our particular application; a comparison of our result to a patch-based approach [43] is shown in Figure 5.

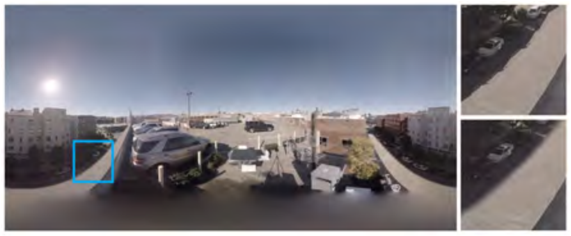

Fig. 5. Left: RGB panorama. Right: Corresponding *inpainted layer* obtained using a patch-based approach [43] (top), and our result using a spring metaphor-based method [5] (bottom). The patch-based approach produces local artifacts that are very prominent when visualized on an HMD. While the spring-metaphor inpainting yields a blurred approximation in the area to inpaint, for our application this smoothness is preferable.

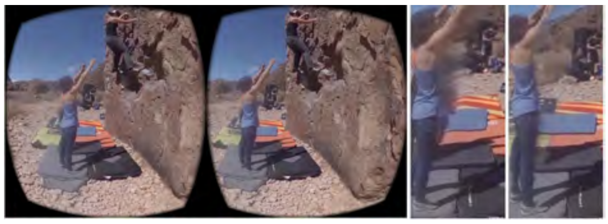

Fig. 6. Comparison of two different methods to modulate the opacity values $\hat{\alpha}^F$ as a function of distance to the center of projection to obtain α^F. Left: Original view as seen in the HMD. Right: Close-up of a displaced view with two different modulations: linear interpolation (left), and blending based on a logistic function (right). Aggravated ghosting results from using a linear interpolation as a function of distance.

3.3 Real-time playback

During playback, the three layers are rendered as meshes. When the viewer is at the center of projection, only the foreground layer should be visible. As they move away from the center, we need to show information from the other layers as well, depending on the disocclusions that take place, as given by the precomputed opacity maps.

To model this behavior, at runtime, we modulate the opacity of each layer given by the opacity maps with a sigmoid function of δ, the current distance between the viewer's head position and the center of projection. Specifically, the run-time foreground layer opacity α^F is given by:

$$\alpha^F = S(\delta)\hat{\alpha}^F + 1 - S(\delta), \tag{3}$$

where $S(\cdot)$ is the function shown in Eq. 2, with $k = 30$ and $c = 0.15$ meters for all scenes tested. The run-time extrapolated layer opacity α^E is obtained in an analogous manner, using its corresponding precomputed opacity map $\hat{\alpha}^E$. Note that, in this way, for $\delta \rightarrow 0$ the opacity of the layers tends to one (i.e., only the foreground layer is visible). As δ increases, the opacity of the layers is given by the opacity values stored in the pre-computed opacity maps. We choose blending with a logistic function (as opposed to using, e.g., a linear interpolation) since overly smooth transitions tend to result in aggravated ghosting, as shown in Figure 6.

4 Depth Improvement for Motion Parallax

This section describes the depth map preprocessing that we perform, in order to improve scene appearance at boundaries.

Our method relies on depth information from the scene. For scenes where a depth map is provided, such as output by a 360° stitching algorithm [1], we can use the provided depth map. Otherwise, we use an off-the-shelf neural network depth estimation algorithm [27].

However, the depth maps provided by existing algorithms are not optimized for reprojection, as also noted by Waechter and colleagues [68]. For example, both stitching accuracy and benchmark scores are relatively insensitive to slight variations in the pixels around object silhouettes, since they are a tiny subset of any depth map. But small errors in the silhouette depths leads to extremely objectionable ragged boundaries when reprojecting to new viewpoints. On the other hand, small depth errors in object interiors are hardly noticeable. Other artifacts include depth bleeding across silhouettes, strong discontinuities in what should be continuous surfaces, and temporal inconsistencies (Figure 7). This is a general problem that applies to all current approaches to depth map estimation.

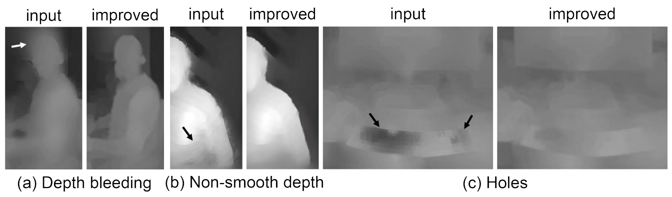

Fig. 7. Examples of artifacts in the input depth videos that hamper the viewing experience, and our resulting improved depth. (a) Bleeding artifacts around object boundaries. (b) Piece-wise discontinuities in what should be smooth gradients. (c) Strong discontinuities in what should be a continuous surface.

Our goal in this section is then not to provide an algorithm that improves the accuracy of the depth maps in general. Instead, we focus on minimizing the three main sources of artifacts, thus making the scene look much more visually plausible and appealing.

We pose improving the input depth map as an optimization problem. The objective function has a data term (E_{data}), and constraints for edge preservation (E_e), spatial smoothness (E_{sm}), and temporal consistency (E_t):

$$\underset{d}{\operatorname{argmin}} \ \lambda_{data}E_{data} + \lambda_e E_e + \lambda_{sm}E_{sm} + \lambda_t E_t, \qquad (4)$$

where d is the depth map of a frame of the video, $d(\theta,\phi)$; in the following, to simplify notation, we will use an index i to denote each (θ,ϕ) pair, that is, each pixel in each equirectangular-format frame. The optimization is solved per frame. We chose the λ values empirically (an evaluation can be found in the supplementary material), for our experiments $\lambda_{data} = 0.1$, $\lambda_e = 1$, $\lambda_{sm} = 0.5$, and $\lambda_t = 0.1$. Note that we pad both RGB and depth maps with the corresponding wrap-around values of the equirectangular projection.

The **data term** ensures fidelity to the input depth values:

$$E_{data}(i) = \sum_i w_d(i)\left(d(i) - \hat{d}(i)\right)^2, \qquad (5)$$

where $\hat{d}$ denotes the input depth. The per-pixel weight $w_d(i)$ models the reliability of the input data. Its aim is to reduce data fidelity along edges, since error in the input depth is more prominent in those regions. It is based on the local variance of the depth σ^2 of each pixel i (the higher its variance, the less reliable). Specifically: $w_d(i) = e^{-\gamma\cdot\sigma^2}$, where γ is set to 10^5 for all cases, and the window size to compute σ^2 is 7×7.

To enforce clean edges, we add an **edge guidance term** that penalizes propagation across edges by using the edge weight $w_e(i,j)$, inspired by the colorization method of Levin et al. [44]:

$$E_e(i) = \sum_i \left(d(i) - \sum_{j\in\mathcal{N}(i)} w_e(i,j)\, d(j)\right)^2, \qquad (6)$$

where j denotes pixels in a neighborhood of pixel i. Unlike Levin et al.'s method, we use $w_e(i,j) = \tau(e(i) - e(j))$, where e represents an edge vote map and τ is Tukey's biweight [7] (comparisons with different functions for τ can be found in the supplementary material). To compute the edge vote map e, we use a multiscale edge detector [22], taking into account edge information from both the RGB image (e_{rgb}) and its corresponding input depth map (e_d), so that $e = e_{rgb} + e_d$. As a result, edges corresponding to a depth difference will have higher edge vote map values, and thus lower weights (w_e).

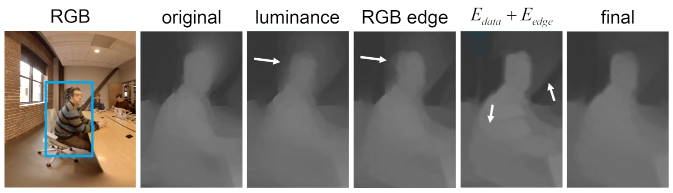

Fig. 8. Depth improvement results for different variations of our optimization. The first and second columns show the input RGB and depth images, respectively. The third and fourth columns show the results using luminance or an edge map computed from RGB as guidance. In the fifth column we show the result when removing the smoothness term (Eq. (7)) from our optimization. As the black arrows indicate, clear artifacts remain in all three results. The last column shows the result of our optimization.

The **smoothness term** acts over local neighborhoods:

$$E_{sm}(i) = \sum_i \sum_{j\in\mathcal{N}(i)} w_{sm}(i)\left(d(i) - d(j)\right)^2, \qquad (7)$$

where w_{sm} is the smoothness weight, obtained in the same way as the data weight w_d, by computing the local variance σ^2 of the input depth map: $w_{sm}(i) = e^{-\beta\cdot\sigma^2}$. In this case σ^2 is computed within a 3×3 neighborhood, and we set $\beta = 10^3$ for all cases. The weight is lower the higher the variance within a local neighborhood, so that smoothness is only imposed in regions where there are no abrupt changes in depth. Figure 8 shows the influence of each term in our improved depth, along with the result using luminance or RGB values.

Finally, **the temporal consistency term** is defined as:

$$E_t(i) = \sum_i w_t(i)\left(d(i) - \psi_{prev\to cur}\left(d_{prev}(i)\right)\right)^2, \qquad (8)$$

where $\psi_{prev\to cur}(\cdot)$ is a warping operator between two frames, implemented as the variational robust optical flow method [47]. The weight $w_t(i)$ is computed as: $w_t(i) = \max\left(\varepsilon, \sqrt{u(i)^2 + v(i)^2}\right)$, where ε is set to 10^{-4}, and $u(i)$ and $v(i)$ correspond to horizontal and vertical flows at pixel i. This weight is higher if there is larger motion, since temporal artifacts would be more noticeable.

Since all the terms are l_2 norms, we solve Eq. (4) with the conjugate gradient method. For time efficiency, we first downscale the input RGB and depth to 80% of the original size in each dimension, then we upscale the refined depth to the original resolution, followed by a fast bilateral filtering [15]. Figure 9 compares our depth refinement step with other common existing approaches; our method provides cleaner depth maps, which are a key factor when adding parallax cues.

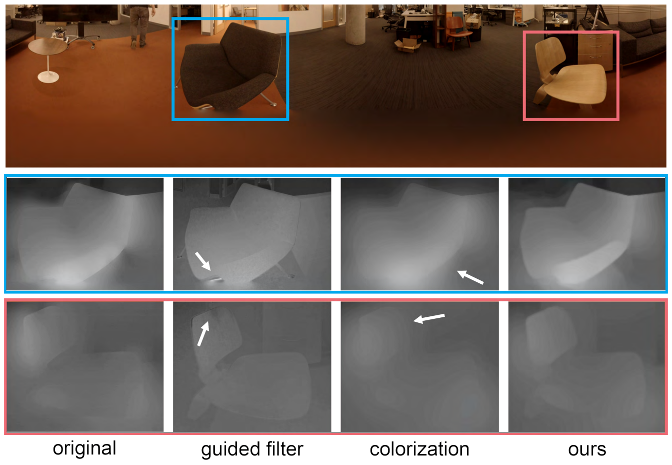

Fig. 9. Comparison of our depth improvement against guided filter [31], and colorization [44]. Since the former is guided by an RGB image, it leads to artifacts similar to those shown in Figure 8, whereas the latter tends to over-smooth the result.

5 Evaluation

Recents studies suggest that 6-DoF provides an overall better viewing experience (e.g., [67]). To validate whether the results achieved with our method also provide an advantage over conventional 3-DoF 360° video viewing, we perform three different user studies using 360° stereo videos with and without motion parallax. Note that for both conditions (3-DoF and 6-DoF) we display stereo views. Specifically, we want to answer two key questions: (a) Does our added motion parallax provide a more compelling viewing experience?, and (b) does it reduce sickness? For the first question on *preference*, we designed two experiments, carried out first and last. In between, we performed the *sickness* experiment. Since the naive handling of disocclusions (Figure 1, close-up (B)) yields very noticeable artifacts, we chose not to include it as a condition in our studies.

All 360° videos were shown on an Oculus Rift connected to a PC equipped with an Nvidia Titan, running at 90fps. Similar to VR applications that use a limited tracking volume, we constrain the maximum displacement from the center of projection using visual cues. To first find what makes a reasonable range of casual, accidental motion, we carried out an initial study in which we registered head movements when watching our 6-DoF 360° content. We define accidental motion

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.

as involuntary head translations that will occur during rotational movements in a natural exploration of the scene. Figure 10 (left) shows the resulting histogram, measuring the Euclidean distance to the center of projection 90 times per second; we observe that most movement is clearly constrained to a certain range. Based on this histogram, we set two thresholds at 20 and 35 cm. When the first threshold is surpassed, blue latitude circles are overlaid to the video content, and the image progressively fades to gray (see Figure 10, right); once the second threshold is surpassed, the image fades to black. For fairness, when comparing between our method and conventional 3-DoF viewing, we added the visual cues for constraining the displacement to both. Participants were informed about this visualization previously to all experiments.

Participants. A total of 24 participants (8 female, 16 male), ages 18 to 20, took part in the experiment. The same subjects participated in the three studies, carried out on three separate days to avoid fatigue and accumulation effects. They voluntarily signed up for our experiments, were naive with respect to their purpose or our technique, and were paid 25$ upon completion of the experiment on the third day. They were first asked to answer a brief questionnaire about their previous use of HMDs and VR content: 14 subjects had no previous experience with HMDs or VR content, 8 subjects had used an HMD less than 3 times before, and 2 subjects had used an HMD up to 10 times before.

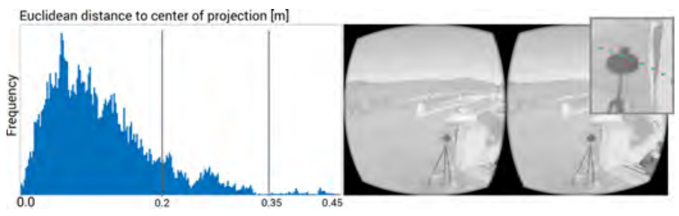

Fig. 10. Left: Histogram of the Euclidean distance from the head position to the center of projection during our pilot experiment. We observe that most movement is limited to a certain range. The two gray lines indicate the two thresholds that control our visualization: at 0.2 m, blue latitude circles appear and the scene progressively fades to gray (shown on the right); at 0.35 m the scene fades to black.

5.1 Experiment #1: Preference (part I)

In the first experiment we evaluate whether our method provides a more compelling viewing experience.

Stimuli. The stimuli consisted of seven 360° videos, covering a variety of scene layouts, content, and motion. For each video, there were two versions: the original one, and adding motion parallax using our method. The videos were captured with a GoPro Odyssey and a Yi Halo cameras, and stitched using the algorithms provided by Google Jump Manager. We show some representative frames in Figure 11, please refer to the supplementary material for more details. To keep the subjects engaged, we limited the videos to 30 seconds each and, following common practice [11, 37], participants were informed that they would be asked a few questions about the scenes after watching them.

Fig. 11. Example frames of the videos used for our user study.

Procedure. Each participant watched the seven videos twice, once without (conventional 3-DoF viewing) and once with motion parallax added with our method. The order was randomized for each video, separated by a one-second black screen. After seeing each pair of videos, they answered a questionnaire where they had to choose one method or the other in terms of realism, comfort, immersion, presence of visual artifacts, and global preference. There was also a space for comments, allowing the participants to explain the reasons for their answer. The whole questionnaire can be found in the supplementary material. At the end of the experiment, a debriefing was carried out, but participants were not told the goal nor any information about the techniques tested in the experiment, since they had to go through two more sessions in the next days. The whole experiment took less than 30 minutes per subject.

Results. We show in Figure 12 (left) the results for the global preference question. The videos with added parallax using our method were preferred in six out of the seven cases. Additionally, seven users commented about the movement and the depth being "more realistic" with our method. Results for the other questions (realism, comfort, immersion, and artifacts) are consistent with the global preference, and can be found in the supplementary material.

5.2 Experiment #2: Sickness

One of the main reasons motivating our technique is the hypothesis that, when watching 360° video, the absence of motion parallax may induce a feeling of sickness or discomfort, due to the mismatch between different sensory inputs (visual and vestibular). In our second experiment, we set out to measure if this is the case, and if it is less prominent when adding motion parallax with our technique.

Stimuli. The stimuli consisted of a set of 12 videos, distinct from those in the previous experiment, but again covering a wide variety of scene layouts and movements. The videos were captured with a GoPro Odyssey and a Yi Halo cameras, and stitched using the algorithms provided by Google Jump Manager. The videos lasted between 30 seconds and one minute each, for a total duration of 8 minutes and 30 seconds.

Procedure. We created two blocks with the set of 360° videos; the first contains the 12 original videos without motion parallax, while the second features motion parallax with our method. There was only a brief pause of 200*ms* between videos, because we wanted to analyze the exposure to a continuous 360° viewing experience. This experiment consisted of two sessions on two different days. On the first session, they watched one block and answered a questionnaire containing (a) the VRSQ questionnaire (Virtual Reality Sickness Questionnaire) [40], and (b) two yes/no questions asking whether they had experienced, at any time during viewing, sickness, dizziness, and/or vertigo, and whether they had experienced discomfort; please refer to the supplementary material for the full questionnaire and details. They were also instructed to describe the video in which they felt such symptoms. On the second, they did the same for the other block. The order of the blocks was randomized between participants. The participants were allowed to stop if they needed to, but none requested to do so, and a debriefing followed the sessions. The whole experiment took 15 minutes per session.

Results. The results of this experiment indicate that our method does indeed help in reducing sickness by enabling motion parallax: while 17 out of the 24 participants reported symptoms of sickness, dizziness, and/or vertigo while watching the videos without motion parallax, only 5 experienced these symptoms with our method. Moreover, none of the subjects reported visual discomfort with our method, while 4 users experienced discomfort with conventional 3-DoF viewing. The results for the VRSQ questionnaire were inconclusive, possibly due to the short duration of the viewing session and the lack of a physical task (the authors of the VRSQ questionnaire [40] report sessions of 90 minutes, and different target selection tasks throughout the session).

5.3 Experiment #3: Preference (part II)

In the last experiment, we repeated the procedure of Experiment #1 but this time disclosing in advance the difference between the two

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.

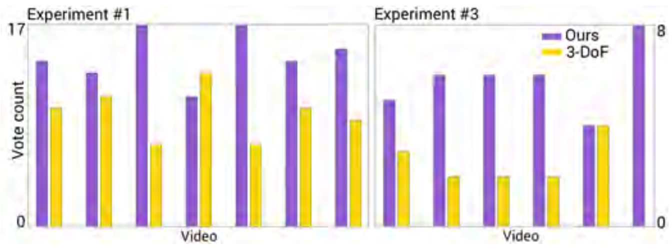

Fig. 12. Vote counts for the global preference question in Experiment #1 (left) and Experiment #3 (right), for our 6-DoF method (purple), and the conventional 3-DoF (orange). The x-axes show the different videos tested. In Experiment #1, the videos with added parallax were preferred in six out of the seven cases, with a strong preference in two cases. In Experiment #3, our method was strongly preferred for five out of the six videos.

versions of each video (motion parallax). The goal was to test if, once the presence of motion parallax is explicitly known, users would find the viewing experience more or less enjoyable than before, and whether this altered their viewing behavior. We first played a video on a conventional desktop display showing a recorded HMD viewing session with and without motion parallax, verbally explaining the difference. Participants then put on the HMD to view the same scene, asking them to experiment moving their head. This process went on until we were sure that the participants understood the differences between the two viewing modes. This scene was only used for the explanation and was not included in the test set. This third experiment was carried out the last day, upon completion of the two previous experiments, to ensure that during the previous sessions they were unaware of the differences between the methods. Note that participants were still not aware that we were testing a new method, only two different viewing options.

Stimuli. Each subject was presented with a random subset of two videos from Experiment #1 (not including the one used during the explanations). Each video was thus viewed and evaluated eight times.

Procedure. After the initial explanations, the procedure was the same as Experiment #1. Since each subject only watched four videos from two scenes, the total duration of this experiment was 10 minutes.

Results. As Figure 12 (right) shows, our method was strongly preferred for five out of the six videos, with no clear preference for the sixth. Furthermore, several participants verbally expressed and confirmed their preference, commenting that our method provided “a more realistic 3D experience”, that it ”greatly helps the feeling of immersion”, and that “the movement is closer to that of the real world”.

5.4 Analysis of viewing behavior

To gain additional insights about the possible influence in viewing behavior of enabling motion parallax, we have analyzed the differences in head movement. For each trial in Experiments #1 and #3, we aggregated the distance from the head to the center of projection across the total viewing time, and performed a dependent t-test to compare the means of the distributions with and without motion parallax. We have found a statistically significant ($t(209) = 2.395, p = 0.018$) difference, indicating that, on average, users displace their head 4.3 cm more every second when 6-DoF are enabled. A possible explanation is that motion parallax allows for a more natural viewing of the scene, and thus fosters exploration. This may be also indirectly influenced by the significant reduction in sickness reported in Experiment #2.

We further analyzed head movement for comparing head movement differences between Experiment #1 and Experiment #3 (i.e., before/after explaining the differences between the methods). We followed the same procedure as in the previous analyses, and we found a statistically significant ($t(83) = 3.484, p = 0.001$) difference in the means. On average, users displaced their head from the center 13.46 cm every second more after knowing the differences between the methods. This would offer an explanation for the difference in preference votes with respect to Experiment #1.

Last, we analyzed separately head movement in Experiment #2, since the nature of this experiment was different from the other two. We followed the same procedure, and found that the results are consistent: there was a statistically significant ($t(275) = 3.352, p = 0.001$) difference, with users moving their head from the center 4.05 cm more with our method than with conventional 3-DoF viewing.

6 Results

We have tested our method in a variety of scenarios providing different kinds of input, including the challenging case of capture systems that do not yield depth maps. This is particularly important, since monocular 360° cameras (e.g., the Ricoh Theta) are more affordable and widespread than more sophisticated camera rigs. In the absence of an input depth map, we first use a Convolutional Neural Network (CNN) to estimate per-frame depth maps from monocular RGB [27]; however, the depth map from the CNN is not of sufficient quality for our purposes, and lacks temporal consistency. Our depth improvement stage significantly increases the quality of the final depth, including temporal coherence, thus enabling motion parallax even from such limited input.

We run the preprocessing of our input RGBD videos in a standard PC equiped with an Intel $i7-3770$ processor (up to 3.90 GHz). The processing time (on average) for our videos (resolution of 2048×1024 pixels) in a single core is: 7.71 seconds per frame for the *depth improvement* step, 762 miliseconds per frame for extracting and processing the *opacity maps*, 319 miliseconds per frame for computing the *extrapolated layer*, and 52.66 seconds (total) for computing the *inpainted layer*. The storage overhead adds to the original RGBD video, two static RGBD images (*extrapolated layer* and *inpainted layer*), a 360° video stream with the *opacity map* corresponding to the *foreground layer*, and an additional image corresponding to the *opacity map* of the *extrapolated layer*. After this processing step, our system runs in real time, providing the 90fps recommended for a satisfactory VR experience.

To test how well our method performs given different input depth maps from different capture systems, this section includes results from the GoPro Odyssey and Yi Halo cameras, both stitched with Google Jump Manager (Figures 1 and 15), from Facebook x24 (Figure 14), and from monocular videos from different sources with estimated depth (Figures 1 and 13). As discussed in Section 4, we remind the reader that the depth maps produced by these methods have not been designed to help generate motion parallax effects; as such, they lead to obvious geometric distortions when generating novel views, due mainly to their ragged edges, the presence of holes, and temporal inconsistencies. As an example, Figure 1 uses Google Jump Manager algorithm for stitching and depth generation: however, distortions are still obvious in the novel views (rightmost scene, insets A and B) before applying our method (inset C). Figure 14 shows a result from Facebook x24: our depth refinement and smooth disocclusion handling method leads to the satisfactory computation of novel views.

Three more results are shown in Figure 15 (more results available in the supplementary material), depicting a variety of scenes. For each result, we show a view from the center of projection, and a displaced view leading to disocclusions. In each of the scenes we highlight regions illustrating the added parallax. Last, we further illustrate the use of layers in Figure 16, where, given a displaced view, we color-code the pixels rendered from each of the three layers.

7 Discussion and Conclusions

We have presented here a technique to enable head motion parallax in 360° video, thus enabling 6-DoF viewing of real-world capture footage. We have designed our method to be independent of a specific hardware, camera setup, or recorded baseline, showing examples from different common 360° capture systems, including depth estimated from scratch by a neural network.

Throughout the paper we assume that each RGB (and depth) frame is a monocular panorama in equirectangular projection, but other projections (such as cube map) are also possible.

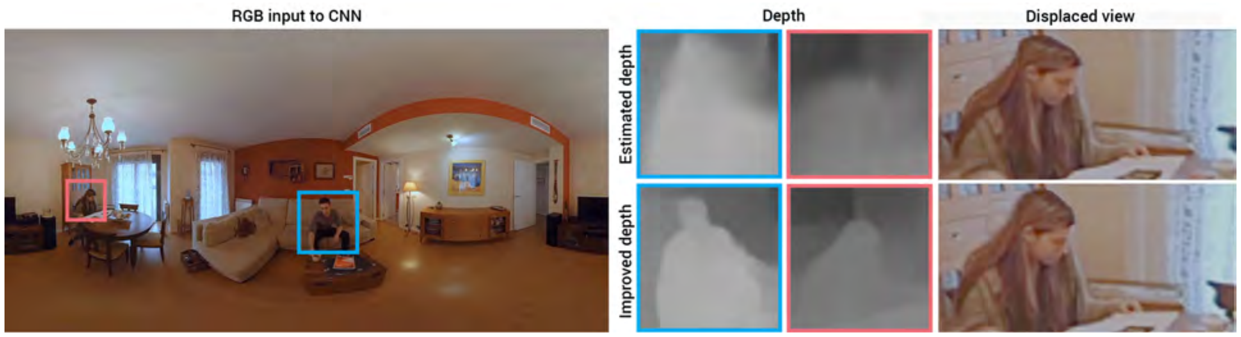

Fig. 13. Result using monocular video without depth as input. We show a representative frame (left), details of the initial estimated depth [27] and our improved depth (middle), as well as the corresponding result when generating a novel view (right). Our method yields minimal distortions even in the presence of such suboptimal input. The scene was taken from a previously existing short clip (dataset from [60]); we use only a 360° RGB monocular view as input and drop the remaining data.

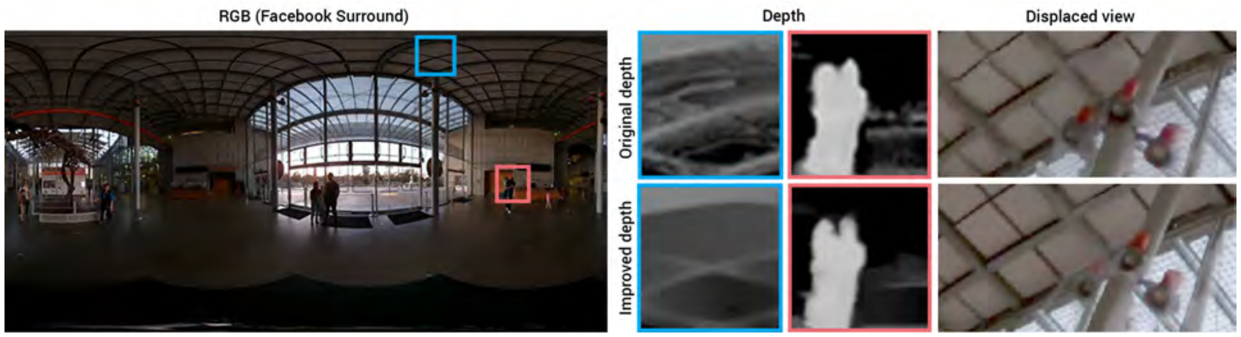

Fig. 14. Left: Representative frame of a video captured by the Facebook x24 camera. Center: Comparison of the original depth provided by Facebook, and our improved depth for the two highlighted regions of the scene. Right: Our improved depth yields better reconstructions of novel views, without distracting artifacts.

Our system requires only RGBD 360° video as input, and is rather robust to inaccuracies in the depth. Thus, it can deal with different data sources, from 360° video plus depth stitched and computed from individual videos from a camera rig, to 360° monocular video with depth computed with a CNN-based depth estimation algorithm. While having access to a multiview setup may not be commonplace, a number of 360° cameras do provide stereo output, which can yield a reasonable depth. Still some common cameras do not provide stereo (Ricoh Theta, Samsung Gear 360, or Nikon Keymission 360, among others), due, e.g., to the limited overlap between the camera views. Our method is designed to be agnostic to the hardware employed during capture, in order not to diminish generality.

Our user studies confirm that our method provides a more compelling viewing experience, while reducing discomfort and sickness. Interestingly, the additional degrees of freedom enabled by our method also influence viewing behavior: On average, users displace more their head from the center of projection when viewing content with 6-DoF.

Limitations and future work. The assumption of a static camera for the input video is reasonable in our scenario, since a considerable amount of 360° content is shot with static cameras. HMD and 360° camera manufacturers typically recommend static cameras [24, 51, 64, 65], and static cameras are widely preferred to moving cameras for most types of 360° videos to reduce potential sickness.

This assumption is mainly required due to the way we compute the extrapolated layer; further, if the camera moved we would need the extrapolated and inpainted layers to be dynamic (i.e., videos instead of images), requiring more storage and bandwidth during playback. Aside from this, our representation could be extended to handle a moving camera, and we leave this to future work.

Our layered representation features three layers. In theory, more than three layers could be needed depending on scene complexity, and this would incur additional storage and processing requirements. The number of moving and/or static objects overlapping in time and space, and in general, the depth complexity of the scene, would need to be assessed together with the increase in algorithmic complexity, in order to choose the optimal number of layers for each scene. In practice, however, we find our solution to be enough, and a number of reasons support our choice: First, three layers represent the types of motion present in many 360° videos: a few moving foreground actors or objects, as well as static objects; second, the amount of storage and bandwidth would increase with the number of layers, eventually hindering real time playback; third, for scenes with greater layer complexity, determining the number and content of layers would be very challenging from 360° video alone, and not necessary for typical videos given the amount of head motion we target, and that has been shown as usual in these setups [67].

Our computation of the extrapolated layer has an implicit limitation for cases with strong lighting variations (e.g., moving shadows). In those cases, depth remains constant but RGB can vary significantly, so we can have some "noise" in the extrapolated layer. In practice, however, the small extent and varying nature of disocclusions result in this effect being negligible. Additionally, there might be some cases in which large scene objects are close to the camera and remain stationary. In such cases, one would need to resort to the *inpainted layer*, and more complex inpainting algorithms may be needed.

Our method relies on the quality of the input depthmap, especially near disocclusion boundaries. We lessen this dependency with our depth improvement step, however, our method does still introduce some artifacts at disocclusion boundaries, which is to be expected given the extremely limited nature of our input. Our studies show that users prefer 6-DoF viewing to 3-DoF viewing, even with these artifacts; getting rid completely of such artifacts remains an open challenge. It is possible that combining ideas from our work and the works by Hedman et al. [32, 33] could lead to higher-quality 6-DoF capture.

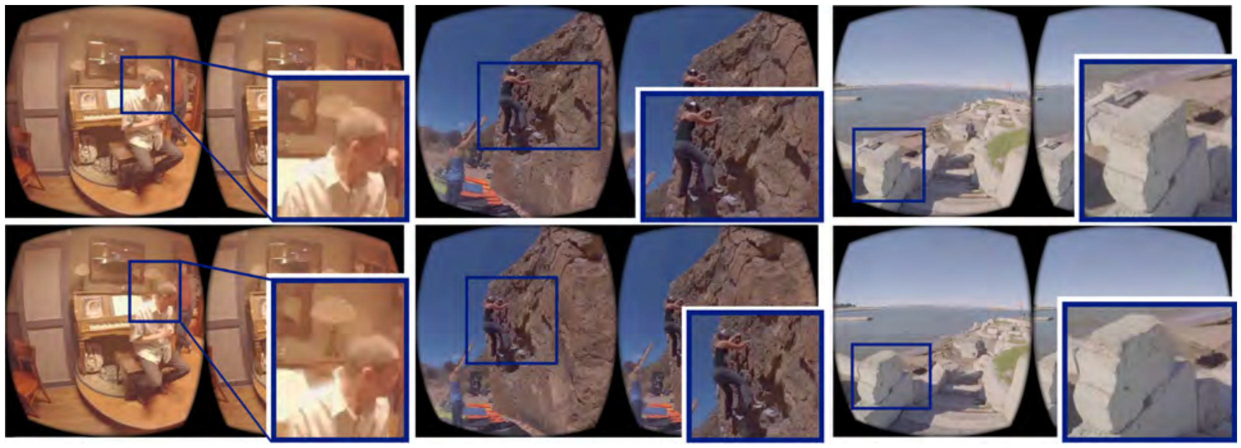

Fig. 15. Three examples of novel views generated with our method. For each example we show the original view (top), and the corresponding displaced view (bottom). We also include close-ups of regions where the added parallax is clearly visible.

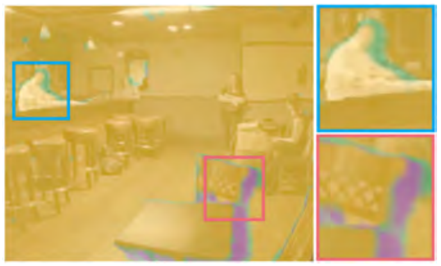

Fig. 16. Color-coded visualization depicting the use of our three-layer representation in a given frame for a displaced view (orange: *foreground layer*; green: *extrapolated layer*; purple: *inpainted layer*).

These errors increase as the viewer moves farther away from the center of projection. In the future, we would like to explore options to minimize this, for instance by creating a non-linear mapping between the head and the camera movement that prevents the viewer from moving too far. Alternatively, existing techniques for controling user attention in VR [17, 30] could be helpful in this context. Last, as a consequence of the omnidirectional stereo (ODS) format, depth information near the poles is not accurate [55], and thus the performance of our method worsens near those regions. However, these regions are rarely observed due to a horizon bias [62].

Many studies exist assessing presence, immersion, or discomfort when exploring virtual reality (synthetic) content. In contrast, existing studies on viewer experience watching 360° footage on HMDs are preliminary and much remains to be explored about how we should display real content on immersive HMDs. We hope that our work provides a solid background for subsequent studies in this area.

Acknowledgments

The authors would like to thank Miguel Crespo for support with experiments, the participants for their colaboration, and Adobe Research for the 360° videos provided (in particular the creators and performers). This research has been partially funded by an ERC Consolidator Grant (project CHAMELEON), and the Spanish Ministry of Economy and Competitiveness (projects TIN2016-78753- P, and TIN2016- 79710-P). Ana Serrano was supported by an FPI grant from the Spanish Ministry of Economy and Competitiveness, an Adobe Research Fellowship, and a Nvidia Graduate Fellowship.

References

[1] R. Anderson, D. Gallup, J. T. Barron, J. Kontkanen, N. Snavely, C. Hernández, S. Agarwal, and S. M. Seitz. Jump: Virtual Reality Video. *ACM Trans. Graph.*, 2016.

[2] A. Atapour-Abarghouei and T. P. Breckon. A comparative review of plausible hole filling strategies in the context of scene depth image completion. *Computers & Graphics*, 72:39–58, 2018.

[3] D. Baricevic, T. Höllerer, and M. Turk. Densification of semi-dense reconstructions for novel view generation of live scenes. In *WACV*, pp. 842–851. IEEE, 2017.

[4] J. T. Barron, A. Adams, Y. Shih, and C. Hernández. Fast bilateral-space stereo for synthetic defocus. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4466–4474, 2015.

[5] M. Bertalmío, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *SIGGRAPH*, pp. 417–424. ACM, 2000.

[6] S. Birchfield and C. Tomasi. Depth discontinuities by pixel-to-pixel stereo. *International Journal of Computer Vision*, 35(3):269–293, 1999.

[7] M. J. Black, G. Sapiro, D. H. Marimont, and D. Heeger. Robust anisotropic diffusion. *IEEE Trans. Image Processing*, 7(3):421–432, 1998.

[8] C. Buehler, M. Bosse, L. McMillan, S. J. Gortler, and M. F. Cohen. Unstructured lumigraph rendering. In *SIGGRAPH*, pp. 425–432. ACM, 2001.

[9] P. Buyssens, M. Daisy, D. Tschumperlé, and O. Lézoray. Depth-aware patch-based image disocclusion for virtual view synthesis. In *SIGGRAPH Asia Technical Briefs*, pp. 2:1–2:4. ACM, 2015.

[10] P. Buyssens, O. L. Meur, M. Daisy, D. Tschumperlé, and O. Lézoray. Depth-guided disocclusion inpainting of synthesized RGB-D images. *IEEE Trans. Image Processing*, 26(2):525–538, 2017.

[11] Z. Bylinskii, P. Isola, C. Bainbridge, A. Torralba, and A. Oliva. Intrinsic and extrinsic effects on image memorability. *Vision research*, 116:165–178, 2015.

[12] T. Chang, Y. Chou, and J. Yang. Robust depth enhancement and optimization based on advanced multilateral filters. *EURASIP J. Adv. Sig. Proc.*, 2017:51, 2017.

[13] G. Chaurasia, S. Duchêne, O. Sorkine-Hornung, and G. Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM Trans. Graph.*, 32(3):30:1–30:12, 2013.

[14] G. Chaurasia, O. Sorkine, and G. Drettakis. Silhouette-aware warping for image-based rendering. *Comput. Graph. Forum*, 30(4):1223–1232, 2011.

[15] J. Chen, S. Paris, and F. Durand. Real-time edge-aware image processing with the bilateral grid. *ACM Trans. Graph.*, 26(3), July 2007. doi: 10.1145/1276377.1276506

[16] A. Criminisi, P. Pérez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Trans. Image Processing*, 13(9):1200–1212, 2004.

[17] F. Danieau, A. Guillo, and R. Doré. Attention guidance for immersive video content in head-mounted displays. In *2017 IEEE Virtual Reality (VR)*, pp. 205–206, 2017.

[18] P. Debevec, G. Borshukov, and Y. Yu. Efficient view-dependent image-based rendering with projective texture-mapping. In *9th Eurographics Rendering Workshop*, 1998.

[19] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH*, pp. 11–20. ACM, 1996.

[20] F. Devernay and A. R. Peon. Novel view synthesis for stereoscopic cinema: Detecting and removing artifacts. In *Proceedings of the 1st International Workshop on 3D Video Processing*, 3DVP '10, pp. 25–30. ACM, New York, NY, USA, 2010. doi: 10.1145/1877791.1877798

[21] P. Didyk, P. Sitthi-Amorn, W. Freeman, F. Durand, and W. Matusik. Joint view expansion and filtering for automultiscopic 3d displays. *ACM Transactions on Graphics (TOG)*, 32(6):221, 2013.

[22] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *ICCV*, pp. 1841–1848. IEEE Computer Society, 2013.

[23] M. Eisemann, B. de Decker, M. A. Magnor, P. Bekaert, E. de Aguiar, N. Ahmed, C. Theobalt, and A. Sellent. Floating textures. *Comput. Graph. Forum*, 27(2):409–418, 2008.

[24] U. Engine. Virtual reality best practices. `https://docs.unrealengine.com/en-us/Platforms/VR/ContentSetup`.

[25] R. S. Feris, R. Raskar, L. Chen, K. Tan, and M. Turk. Discontinuity preserving stereo with small baseline multi-flash illumination. In *ICCV*, pp. 412–419. IEEE Computer Society, 2005.

[26] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[27] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, pp. 6602–6611. IEEE Computer Society, 2017.

[28] M. Goesele, J. Ackermann, S. Fuhrmann, C. Haubold, R. Klowsky, D. Steedly, and R. Szeliski. Ambient point clouds for view interpolation. *ACM Trans. Graph.*, 29(4):95:1–95:6, 2010.

[29] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *SIGGRAPH*, pp. 43–54. ACM, 1996.

[30] S. Grogorick, M. Stengel, E. Eisemann, and M. Magnor. Subtle gaze guidance for immersive environments. In *Proceedings of the ACM Symposium on Applied Perception*, SAP '17, pp. 4:1–4:7, 2017.

[31] K. He, J. Sun, and X. Tang. Guided image filtering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(6):1397–1409, 2013.

[32] P. Hedman, S. Alsisan, R. Szeliski, and J. Kopf. Casual 3d photography. *ACM Trans. Graph.*, 36(6):234:1–234:15, 2017.

[33] P. Hedman and J. Kopf. Instant 3d photography. *ACM Trans. Graph.*, 37(4):101:1–101:12, July 2018.

[34] A. T. Hinds, D. Doyen, and P. Carballeira. Toward the realization of six degrees-of-freedom with compressed light fields. In *ICME*, pp. 1171–1176. IEEE Computer Society, 2017.

[35] J. Huang, Z. Chen, D. Ceylan, and H. Jin. 6-dof VR videos with a single 360-camera. In *VR*, pp. 37–44. IEEE Computer Society, 2017.

[36] Intel and H. Technology. Hypevr. `https://hypevr.com/`, 2018. Last accessed June 1st, 2018.

[37] T. Judd, K. Ehinger, F. Durand, and A. Torralba. Learning to predict where humans look. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[38] N. K. Kalantari, T. Wang, and R. Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Trans. Graph.*, 35(6):193:1–193:10, 2016.

[39] P. Kellnhofer, P. Didyk, S.-P. Wang, P. Sitthi-Amorn, W. Freeman, F. Durand, and W. Matusik. 3dtv at home: Eulerian-lagrangian stereo-to-multiview conversion. *ACM Trans. Graph.*, 36(4):146:1–146:13, July 2017.

[40] H. K. Kim, J. Park, Y. Choi, and M. Choe. Virtual reality sickness questionnaire (vrsq): Motion sickness measurement index in a virtual reality environment. *Applied Ergonomics*, 69:66–73, 2018.

[41] B. Koniaris, M. Kosek, D. Sinclair, and K. Mitchell. Real-time rendering with compressed animated light fields. In *Graphics Interface*, pp. 33–40. Canadian Human-Computer Communications Society / ACM, 2017.

[42] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele. Joint bilateral upsampling. *ACM Trans. Graph.*, 26(3):96, 2007.

[43] J. H. Lee, I. Choi, and M. H. Kim. Laplacian patch-based image synthesis. In *CVPR*, pp. 2727–2735. IEEE Computer Society, 2016.

[44] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM Trans. Graph.*, 23(3):689–694, 2004.

[45] A. Levin, D. Lischinski, and Y. Weiss. A closed-form solution to natural image matting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(2):228–242, 2008.

[46] C. Lipski, C. Linz, K. Berger, A. Sellent, and M. Magnor. Virtual video camera: Image-based viewpoint navigation through space and time. *Computer Graphics Forum*, 29(8):2555–2568, Dec 2010.

[47] C. Liu. *Exploring new representations and applications for motion analysis*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2009.

[48] B. Luo, F. Xu, C. Richardt, and J. Yong. Parallax360: Stereoscopic 360° scene representation for head-motion parallax. *IEEE Trans. Vis. Comput. Graph.*, 24(4):1545–1553, 2018.

[49] D. Mahajan, F.-C. Huang, W. Matusik, R. Ramamoorthi, and P. Belhumeur. Moving gradients: A path-based method for plausible image interpolation. *ACM Trans. Graph.*, 28(3):42:1–42:11, July 2009.

[50] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pp. 39–46, 1995.

[51] Oculus. Oculus introduction to best practices. `https://developer.oculus.com/design/latest/concepts/book-bp/`, 2017. Last accessed September 10th, 2018.

[52] R. S. Overbeck, D. Erickson, D. Evangelakos, M. Pharr, and P. Debevec. A system for acquiring, compressing, and rendering panoramic light field stills for virtual reality. *ACM Trans. Graph.*, 37, 2018.

[53] E. Penner and L. Zhang. Soft 3d reconstruction for view synthesis. *ACM Trans. Graph.*, 36(6):235:1–235:11, 2017.

[54] J. Philip and G. Drettakis. Plane-based multi-view inpainting for image-based rendering in large scenes. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '18, pp. 6:1–6:11, 2018.

[55] C. Richardt, Y. Pritch, H. Zimmer, and A. Sorkine-Hornung. Megastereo: Constructing high-resolution stereo panoramas. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1256–1263, June 2013.

[56] T. Rongsirigul, Y. Nakashima, T. Sato, and N. Yokoya. Novel view synthesis with light-weight view-dependent texture mapping for a stereoscopic HMD. In *ICME*, pp. 703–708. IEEE Computer Society, 2017.

[57] E. Sayyad, P. Sen, and T. Höllerer. Panotrace: interactive 3d modeling of surround-view panoramic images in virtual reality. In *VRST*, pp. 32:1–32:10. ACM, 2017.

[58] C. Schroers, J.-C. Bazin, and A. Sorkine-Hornung. An omnistereoscopic video pipeline for capture and display of real-world vr. *ACM Trans. Graph.*, 37(3):37:1–37:13, Aug. 2018.

[59] S. M. Seitz and C. R. Dyer. View morphing. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pp. 21–30, 1996.

[60] A. Serrano, V. Sitzmann, J. Ruiz-Borau, G. Wetzstein, D. Gutierrez, and B. Masia. Movie editing and cognitive event segmentation in virtual reality video. *ACM Transactions on Graphics (SIGGRAPH 2017)*, 36(4), 2017.

[61] Q. Shan, B. Curless, Y. Furukawa, C. Hernández, and S. M. Seitz. Occluding contours for multi-view stereo. In *CVPR*, pp. 4002–4009. IEEE Computer Society, 2014.

[62] V. Sitzmann, A. Serrano, A. Pavel, M. Agrawala, D. Gutierrez, B. Masia, and G. Wetzstein. How do people explore virtual environments? *IEEE Transactions on Visualization and Computer Graphics*, 2017.

[63] N. Stefanoski, O. Wang, M. Lang, P. Greisen, S. Heinzle, and A. Smolic. Automatic view synthesis by image-domain-warping. *IEEE Trans. Image Processing*, 22(9):3329–3341, 2013.

[64] J. Studios. The cinematic vr field guide - a guide to best practices for shooting 360. `https://www.jauntvr.com/cdn/uploads/jaunt-vr-field-guide.pdf`.

[65] U. Technologies. Movement in vr. `https://unity3d.com/es/learn/tutorials/topics/virtual-reality/movement-vr`.

[66] J. Thatte, J. Boin, H. Lakshman, and B. Girod. Depth augmented stereo panorama for cinematic virtual reality with head-motion parallax. In *ICME*, pp. 1–6. IEEE Computer Society, 2016.

[67] J. Thatte and B. Girod. Towards perceptual evaluation of six degrees of freedom virtual reality rendering from stacked omnistereo representation. *Electronic Imaging*, 2018(5):352–1–352–6, 2018.

[68] M. Waechter, M. Beljan, S. Fuhrmann, N. Moehrle, J. Kopf, and M. Goesele. Virtual rephotography: Novel view prediction error for 3d reconstruction. *ACM Trans. Graph.*, 36(4), Jan. 2017.

[69] T.-C. Wang, J.-Y. Zhu, N. K. Kalantari, A. A. Efros, and R. Ramamoorthi. Light field video capture using a learning-based hybrid imaging system. *ACM Trans. Graph.*, 36(4):133:1–133:13, July 2017.

[70] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Trans. Graph.*, 37(4):65:1–65:12, July 2018.

[71] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. A. J. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. *ACM Trans. Graph.*, 23(3):600–608, 2004.